慶應義塾大学学術情報リポジトリ
Keio Associated Repository of Academic resouces

Title	Eメールシステムを用いた人との繋がりの可視化研究
Sub Title	Visualization Study of Precious Relationship Between People Using Email System
Author	松岡, 慧(Matsuoka, Kei) 小木, 哲朗(Ogi, Tetsuro)
Publisher	慶應義塾大学大学院システムデザイン・ マネジメント研究科
Publication year	2013
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2013年度システムデザイン・ マネジメント学 第147号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002013-0060

慶應義塾大学学術情報リポジトリ(KOARA)に掲載されているコンテンツの著作権は、それぞれの著作者、学会または出版社/発行者に帰属し、その権利は著作権法によって保護されています。引用にあたっては、著作権法を遵守してご利用ください。

The copyrights of content available on the KeiO Associated Repository of Academic resources (KOARA) belong to the respective authors, academic societies, or publishers/issuers, and these rights are protected by the Japanese Copyright Act. When quoting the content, please follow the Japanese copyright act.

修士論文　　2013 年度

E メールシステムを用いた
人との繋がりの可視化研究

松岡 慧
（学籍番号：81233636）

指導教員　小木 哲朗

2014 年 3 月

慶應義塾大学大学院システムデザイン・マネジメント研究科
システムデザイン・マネジメント専攻

論　　文　　要　　旨

学籍番号	81233636	氏　名	松岡　慧

論 文 題 目：

E メールシステムを用いた人との繋がりの可視化研究

（内容の要旨）

近年の科学技術の発展により，情報通信機器が我々の生活に広く普及した．しかしながら，それらの普及にも関わらず，一人暮らしの増加などにより人間関係の希薄化が進み，人との繋がりの不足が懸念されている．この背景を受け，本研究では，最も一般的に普及している E メールシステムに注目し，そのシステムにおける送受信履歴を表示する E メールシステムの開発を試みた．本システムは，その表示により，人との繋がりの状況に対する気づきを与えることで，その継続・深化に向けたユーザ行動への誘導を期待するものである．

本研究では，まず，週，月，年などの各タイムスケールを総合的に考えるマルチタイムスケールの視点からデザイン展開を行っている．その結果，送受信先の各アイコンをセルの集合体で表現し，セルオートマトンを用いることで，そのアイコンが送受信頻度に伴いセルの数，形状，および透明度が変化する 6 つの機能を有する E メールシステムをデザインしている．これらの機能により，人との繋がりを示唆できることを狙いとしたものである．さらに，その 1 次プロトタイプを制作し，それを用いた官能評価実験を実施している．その結果，企画性，機能性などすべての評価が高評価を得たものの，そのなかでも操作性の評価が 3.53 点と比較的低いことが判明している．

そこで，次に，本 E メールシステムの操作性向上のために，各アイコンの表示に，送受信先の名前と，ここ 1 年間，1 か月間，および 1 週間の送受信数を加えて提示するリデザインを行っている．そして，その 2 次プロトタイプを制作し，1 次プロトタイプと同様に官能評価実験を実施した．その結果，操作性の評価が 3.53 点から 4.33 点へと大幅に向上するとともに，企画性，機能性，意匠性の評価も向上していることが示された．本 E メールシステムの特徴は，アイコンの変化により E メールの送受信頻度を可視化することで人との繋がりを示唆する 6 つの機能である．この人との繋がりを示唆する 6 つの機能により，人との繋がりへの気づきを与え，コミュニケーションを促進させることを狙いとしている．

以上より，本研究では，E メールの送受信頻度を可視化することで送受信先との繋がりの状況への気づきを促す E メールシステムを開発し，人との繋がりの充足の一助としている．

キーワード（5 語）

E メール，人との繋がり，可視化，アイコン，送受信数

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81233636	Name	Kei Matsuoka

Title

Visualization Study of Precious Relationship Between People Using Email System

Abstract

With the development of science and technology in recent years, information and communication devices have been widely used in our lives. However, despite the spread of these devices, there are growing concerns about the lack of connections between people as they are less and less associated with others for reasons such as the increase in people living alone. With these backgrounds, this study focuses on the most common device, Email systems, and attempts to design an Email system that visualizes the history of frequency in sending and receiving Emails. This system aims to encourage sustainable and deep communication by making people realize the condition of precious relationship between people with the visualization.

This study develops the design from the viewpoint of multi-time scale that comprehensively deals with various time scales such as a week, a month, and a year. As a result, with expressing each icon for senders and receivers of Emails as an aggregate of cells and using cellular automaton, the Email system is designed to have six functions that an icon changes its number of cells, form, and transparency depending on the frequency of sending and receiving Emails. This system aims to indicate the condition of relationship between people using these functions. Moreover, the first prototype is produced and the sensory evaluation experiment with the prototype by the five-step SD method is conducted. The experiment shows that although all the evaluations, such as inventiveness and functionality, obtained high evaluation points, the evaluation for usability is comparatively low as 3.53 points.

Next, toward the improvement in the usability of this Email system, the name of senders and receivers and each number of Emails sent and received in three time scales, this week, this month, and this year, are added and visualized for each icon display. The second prototype is then produced and the sensory evaluation experiment in the same manner as the first prototype is conducted. As a result, the evaluations for inventiveness, functionality, and comprehensive evaluation are also improving, while the evaluation for usability improves greatly to 4.33 points from 3.53 points.

Thus, this study is contributory to fulfill the precious relationship between people, as it develops the Email system encouraging people to realize the condition of relationship between senders and receivers by visualizing the frequency of Emails.

Key Word(5 words)

Email, Precious Relationship Between People, Visualization, Icon, Frequency of Send and Receive

目次

第 1 章　緒言 ····· 1
1.1 研究の背景 ····· 2
1.2 研究の目的と方法 ····· 4
1.3 本論文の構成 ····· 5

第 2 章　1 次デザイン展開：変化するアイコンのデザイン ····· 6
2.1　1 次デザイン展開 ····· 7
2.1.1　デザイン要素の抽出 ····· 7
2.1.2　デザイン要素の分類と機能の導出 ····· 9
2.1.3　デザイン要素の構造化とシステムパラメータの詳細化 ····· 11
2.2　1 次デザイン解 ····· 13
2.2.1　E メールシステムの概要 ····· 13
2.2.2　GUI ····· 15
2.2.3　アイコンによる 6 つの機能 ····· 16
2.3　1 次プロトタイプの評価 ····· 24
2.3.1　官能評価実験の概要 ····· 24
2.3.2　実験結果の解析とその考察 ····· 25
2.3.3　1 次プロトタイプ全体の考察 ····· 27

第 3 章　2 次デザイン展開：送受信数履歴の表示 ····· 30
3.1　2 次デザイン展開 ····· 31
3.1.1　デザイン要素の抽出 ····· 31
3.1.2　デザイン要素の構造化 ····· 32
3.2　2 次デザイン解 ····· 35
3.2.1　送受信先の名前表示機能 ····· 35
3.2.2　送受信数の履歴表示機能 ····· 36
3.3　2 次プロトタイプの評価 ····· 37
2.3.1　官能評価実験の概要 ····· 37
2.3.2　実験結果の解析とその考察 ····· 38
2.3.3　2 次プロトタイプ全体の考察 ····· 40

第 4 章　結言 ····· 41
4.1　本研究の成果 ····· 42
4.2　今後の課題 ····· 43

謝辞 ····· 44
参考文献 ····· 45

Appendix ····· 46
A.1　1 次プロトタイプ　ソースコード ····· 46
A.2　2 次プロトタイプ　ソースコード ····· 80

図目次

1-1　モノの豊かさから心の豊かさへの変化 ……3
1-2　本論文の構成 ……5
2-1　セルラ・オートマトン ……8
2-2　1 次デザイン展開における要素間関係図 ……12
2-3　E メールシステムの操作画面……13
2-2　GUI に関する要素間関係図 ……15
2-5　機能 1：人との繋がりの深さを示唆するアイコンの拡大 ……16
2-6　機能 2：人との繋がりの深さの変化を示唆する透明度変化 ……17
2-7　機能 3：繋がりの薄さを示唆するアイコンの縮小 ……18
2-8　機能 4：人との安定した繋がりを示唆するアイコンの成長傾向の変化 ·19
2-9　機能 5：人との繋がりの成長段階を示唆する形状特徴の変化 ……20
2-10　機能 6：人との繋がりの広さを示唆する新規アイコンの生成……21
2-11　各機能のシステムフロー図 ……23
2-12　各評価項目の評価点平均 ……26
2-13　導出した E メールシステムによる「繋がり」の深化の過程……29
3-1　2 次デザイン展開における要素間関係図 ……34
3-2　2 次デザイン解の操作画面 ……35
3-3　各評価項目の評価点平均 ……39

表目次

1-1　主な情報通信機器の世帯保有状況 …… 3
2-1　6 つの機能における機能と効果 …… 22
2-2　質問項目 …… 24
3-1　質問項目 …… 37

第 1 章

緒言

1.1 研究の背景

近年の科学技術の発展により，大量生産-大量消費型の社会が実現された．これにより，携帯電話やスマートフォンなどの安価かつ高性能な情報通信機器が，我々の生活に広く普及した. 表 1-1 に主な情報通信機器の世帯保有状況を示す. 平成 19 年からすでに携帯電話・PHS は 9 割以上の世帯が保有していることがわかる．また，スマートフォンに関しては平成 22 年では保有している世帯が 1 割程度であったが，平成 23 年には保有している世帯が 3 割まで増加しており，急速に我々の生活に普及していることがわかる．このような情報通信機器の普及に伴い，物質的な豊かさが我々の生活に充足されたといえる.

しかしながら，図 1-1 に示すように，近年，物質的な面で生活を豊かにすることに重きをおきたいと考える人の割合よりも，心の豊かさやゆとりのある生活をすることに重きをおきたいと考える人の割合が 3 割以上多いことがわかる. このことから，効率性や利便性といった物質的な豊かさを重視する価値観から，精神的な豊かさを重視する価値観へ移行しているといえる．この価値観の移行を受け，利便性の高いコミュニケーションツールである情報通信機器の普及により物質的な豊かさは充足されながらも，一人暮らしや核家族化の増加や高齢者の孤独死など「人との繋がり」の希薄化が社会問題として指摘されている. この傾向は，特に，東日本大震災が発生した 2011 年以降に顕著であることが伺える．しかしながら，このような状況下,「人との繋がり」の希薄化に対応する情報通信機器は未だ存在していない．そのため,「人との繋がり」の希薄化は，今日における情報通信機器の課題の 1 つであるといえる．もし，情報通信機器が，送受信数が少なく,「人との繋がり」が希薄化している状態への気づきを与える場を提供できる機能を有していれば，上記の問題への対応になる可能性があり，そのような情報通信機器の開発が望まれていると考える.

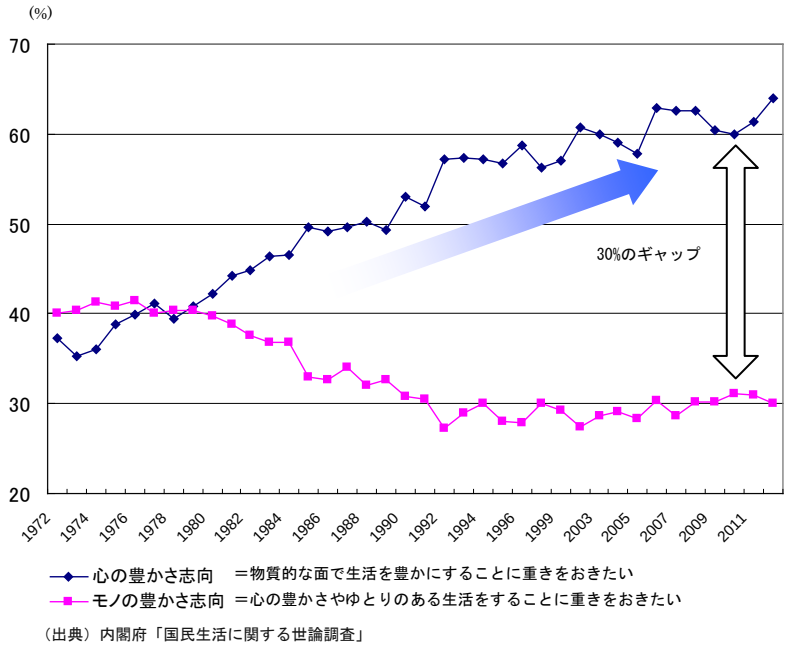
表 1-1 主な情報通信機器の世帯保有状況（平成 19～23 年末）

（%）

	平成19年末（n＝3,640）	平成20年末（n＝4,515）	平成21年末（n＝4,547）	平成22年末（n＝22,271）	平成23年末（n＝16,530）
携帯電話・PHS（スマートフォン含む）	95.0	95.6	96.3	93.2	94.5
固定電話	90.7	90.9	91.2	85.8	83.8
パソコン	85.0	85.9	87.2	83.4	77.4
FAX	55.4	53.5	57.1	43.8	45.0
インターネットに接続できるテレビ	11.7	15.2	23.2	26.8	33.6
インターネットに接続できる家庭用ゲーム機	15.2	20.8	25.9	23.3	24.5
タブレット型端末				7.2	8.5
その他インターネットに接続できる家電（情報家電）等	4.3	5.5	7.6	3.5	6.2
（再掲）スマートフォン				9.7	29.3

n：世帯数

（出典）総務省「平成 23 年通信利用動向調査」



（出典）内閣府「国民生活に関する世論調査」

図 1-1 モノの豊かさから心の豊かさへの変化

1.2 研究の目的と方法

研究の背景から，我々の生活に普及した情報通信機器を積極的に活用し，人との繋がりという精神的な豊かさの充足を促すコンテンツやプロダクトが求められていると考えられる．そこで，本研究では，最も一般的に普及している E メールに注目し，そのシステムにおける送受信数の履歴を可視化する E メールシステムのデザインを試みた．本システムは，その可視化により，人との繋がりの状況に対して，ユーザに気づきを与え，その継続・深化に向けたコミュニケーションを促進することを狙いとした．

研究の方法としては，まず，週単位，月単位，年単位などのさまざまなタイムスケールを総合的に取り扱うマルチタイムスケールの視点から，デザイン展開を進めることにした．これにより，各タイムスケールでの送受信数の履歴を示すことで，単に送受信数の頻度のみならず，その増加・減少傾向（時間微分）や累積（時間積分）示唆することが可能になる．さらに，使えば使うほど，ユーザがそのモノに愛着を持ち，長期的に価値が成長する価値成長モデルを視点として加えることにした．

この 2 つを視点として，まず，1 次デザイン展開を行うことにした．具体的には，関連するデザイン要素の抽出，分類，および構造化を行いつつ，アイデアを模索し，1 次デザイン解を求める．次に，それにより得られたデザイン解の 1 次プロトタイプを製作し，企画性，機能性，操作性，意匠性などに関して官能評価実験を実施し，その有効性を確認する．

さらに，その官能評価実験の結果により得られた低い評価項目に対して改善を行うべく，再度，2 次デザイン展開を行い，得られた 2 次デザイン解の 2 次プロトタイプを制作する．そして，その 2 次プロトタイプを用いて官能評価実験を行うことで，その有効性を確認する．それらにより，デザインされた E メールシステムの，人との繋がりの継続・深化に向けた効果についての考察を行う．

1.3 本論文の構成

本論文の構成を図 1-2 に示す．2 章では，1 次デザイン展開を通じて 1 次デザイン解である E メールシステムを導出し，その 1 次プロトタイプを用いた有効性の官能評価実験を実施する．3 章では，2 章で導出されたデザイン解の課題をさらに改善すべく，2 次デザイン展開を行い，そのプロトタイプを用いて，再度，官能評価実験により検証する．4 章では，本研究の成果と今後の課題を述べ，本研究を総括する．

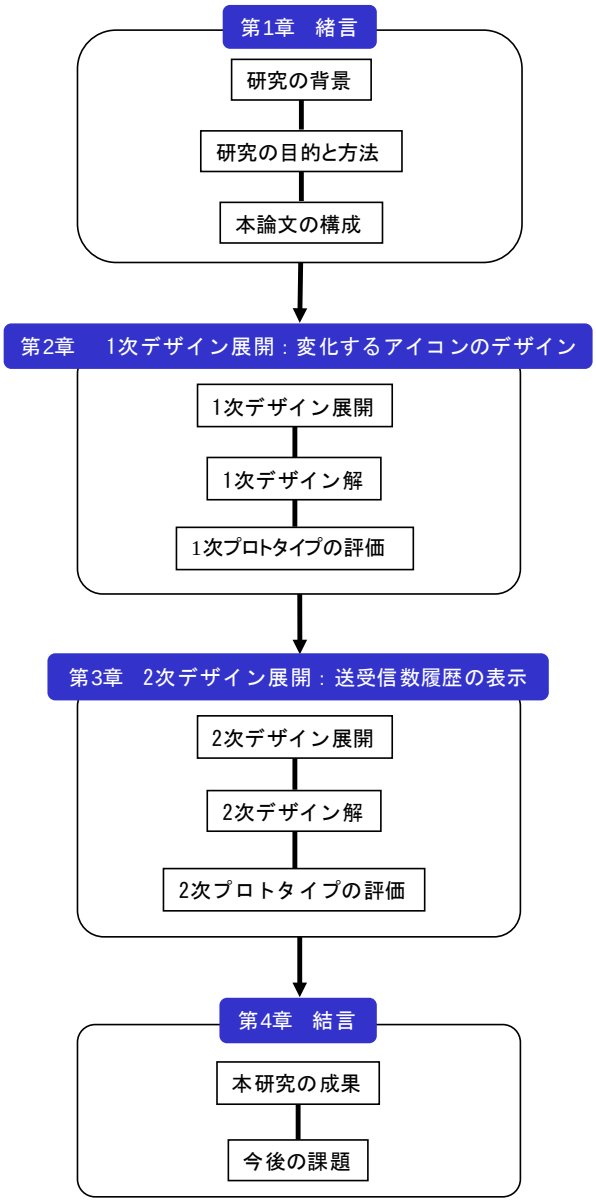


図 1-2 本論文の構成

第 2 章

1 次デザイン展開 :
変化するアイコンのデザイン

2.1　1 次デザイン展開

2.1.1 デザイン要素の抽出

人との繋がり（以下，繋がり）の状況に対する気づきを与え，その継続・深化に向けたユーザ行動への誘導を狙いとし，どのような価値や意味が必要かを念頭に置き，デザイン要素の抽出を行った．まず価値として，デザインの目標となる「繋がりの気づき」と「繋がりの継続・深化への誘導」という価値要素が抽出され，それに付随する価値要素として，「繋がりを深められる喜び」「かけがえのない関係を築ける満足感」などが抽出された．次に，意味として，「繋がりへの気づき」に関連するイメージや機能として「繋がりを実感する」「繋がりを成長する」「繋がりを定着する」といった意味要素が抽出された．また，状態と属性におけるデザイン要素は，状態においては，「E メールの送受信頻度」などが抽出された．属性においては，E メールシステムを使用するデバイスや技術シーズである多様解導出システムなどの属性要素が抽出された．

その後，抽出されたデザイン要素を整理し，繋がりとの関係性の導出を試みた．まず意味要素として抽出された「目に見えない繋がりを意識できる」という要素に着目した．繋がりの継続・深化を実現する上では，ユーザが繋がりを意識できること，すなわち目に見えない繋がりを意識できるようにすることが重要である．そこで人間関係に関連する状態量を可視化することで繋がりの状態を示唆する必要があると考えた．具体的な人間関係に関連するパラメータとして，状態要素である「E メールの送受信頻度」に着目した．E メールの送受信頻度はコミュニケーションの頻度と考えられるため，間接的に人間関係の状態を表す状態量であるといえる．

以上から「目に見えない繋がりを意識できる」という意味要素と「E メールの送受信頻度」という状態要素を関連付け，「E メールの送受信頻度を可視化することで，目に見えない繋がりの状態を示唆する」という繋がりと E メールの関係性を導出することができた．

さらに，E メールの送受信頻度を可視化する具体的な方法として，属性要素として抽出した多様解導出システムに注目し，登録相手を表現するアイコンを生成し，そのビジュアルの変化を用いることを発想した．

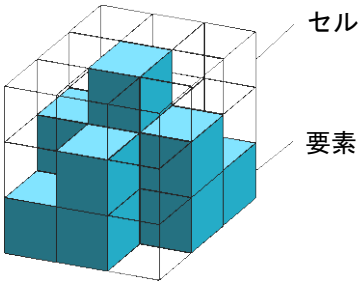


図 2-1　セルラ・オートマトン

ここでは，同 E メールシステムのアイコン形状の特徴変化に用いた多様解導出システムについて説明する．多様解導出システムは，セルラ・オートマトン（以下 CA と略記する）を用いて自己組織的に多様な形状の生成を行うシステムである[1].

CA は，図 2-1 に示すように，セルの状態 S_i，時間ステップ t におけるセルの状態 $C_{[t]}$，時間ステップ t における近傍セルの状態 $N_{[t]}$を入力として，状態遷移関数 f（状態遷移ルール）を用いることで時間ステップ t+1 におけるセルの状態 $C^{[t+1]}$が出力として決定される[2,3]．式では次式のように表される．

$$C^{[t+1]} = f\left(C^{[t]}, N^{[t]}\right) \qquad (1\text{-}1)$$
$$C = \left\{S_1, S_2, \cdots S_{n-1}, S_n\right\}$$

なお，式(1-1)の下段の式は，S_n のなかの 1 つの状態を C が持つことを表現している．同システムでは，ボクセルの要素を用いて CA におけるセルを表現している．そのため，セルの状態を表す S_i は要素の有りと無しの 2 種類であり，時間ステップ t における注目要素と近傍要素の状態から決まる状態遷移ルールを用いることで，時間ステップ t+1 において発生する要素を決定する．なお，同システムでは，3 次元のボクセルを用いて CA における要素を表現している．

以上を通じて，価値要素，意味要素，状態要素および属性要素という 4 つのデザイン要素を整理することで，繋がりと E メールシステムとの関係性を理に適った形で導出することができた．

2.1.2 デザイン要素の分類と機能の導出

2.1.1 節において発想した繋がりと E メールシステムの関係性を具体化するためにデザイン要素を分類した．また，デザイン要素が不足していた場合も 2.1.1 と同様に抽出した．

まず，価値空間において価値成長モデルを導入し，「繋がりの継続・深化」の過程を 5 つの期間で表現した．そして，各期間に必要な価値要素を分類し，不足している価値要素を抽出した．具体的には，価値発見期では「新しいメールシステムへの期待」，価値実感期では「アイコンの成長への驚き」，価値成長期では「繋がりを深められる喜び」，価値定着期では「かけがえのない繋がりを築いた満足感」，および価値伝承期では「自身のコミュニケーションの履歴を残せる安心感」といった価値要素が分類された．

次に，意味空間において価値成長モデルを導入し，5 つの各期間において関連する価値要素との関係を意識しながら意味要素を分類した．さらにマルチタイムスケール・モデルを導入し，各期間で分類した意味要素を，その要素が属すると考えられるタイムスケールで分類した．例として，価値定着期では「かけがえのない繋がりを築いた満足感」という価値要素との関係を意識し，「繋がりが安定したことを示唆する」という意味要素を分類した．また繋がりが安定した状態は，年，月単位での交流を経た状態であると考えられるので「繋がりが安定したことを示唆する」という意味要素をロングタイム・スケールに分類した．

最後に，分類した意味要素が表現する繋がりの状態をアイコンと関連付けるため，繋がりの表現に用いる状態量として，アイコンに関する 3 つのパラメータを抽出した．アイコンの要素数，アイコンの透明度，アイコンを構成する要素の形状特徴の 3 つである．順に，ショートタイム・スケール，ミディアムタイム・スケール，およびロングタイム・スケールに対応している．

マルチタイムスケール・モデルおよび価値成長モデルを視点として価値要素および意味要素を分類したことで，各要素がどのタイムスケールにおいてどのように繋がりの継続・深化と関連するのかが容易に把握することが可能となった．そして，それらの意味要素の関係性をまとめ，アイコンに設定した 3 つの

パラメータと関連付けた.

以上より，E メールの送受信頻度をアイコンによって可視化し繋がりを示唆する 6 つの機能を具体的に導出することができた.

2.1.3 デザイン要素の構造化とシステムパラメータの詳細化

2.1.2 節で得られたシステムの具体的な構想をシステムとして構築するため，デザイン要素の構造化を行った．この構造化の際には，E メールの送受信頻度とその時間経過を可視化する機能（意味要素）と繋がりの表現に用いるアイコンの 3 つのパラメータ（状態要素）との関係性を重点的に設定した．具体的には，意味空間と状態空間を統合し，意味要素によって生じる「アイコンの要素数」，「アイコンの透明度」，および「アイコンを構成する要素の形状特徴」の 3 つのパラメータの変化を具体的な数値で設定した．最終的に得られた要素間関係図を図 2-2 に示す[4].

本研究におけるデザイン要素の構造化では，意味空間と状態空間を統合し，意味要素による状態要素の変化の記述とその具体的な数値の変化の記述を同時に設定した．具体例としては，機能「繋がりの深さを示唆するアイコンの成長」が，条件「メールを送受信した」を満たしたときパラメータ「要素数」が 1~10 個増加するというように設定した．このように詳細に設定することで，実際の変化がパラメータの数値の変化で記述され，意味要素だけでは想定することが困難だった実際の変化を容易に理解し，共有できるようになった．また，価値成長モデルやマルチタイムスケール・モデルに沿った継続的なパラメータの変化も理解でき，実際にプログラムを用いてシステムを構築する際の効率を上げることができた[5,6].

以上のようにデザイン要素の構造化により，最終的な E メールシステムのデザインを詳細に決定することが可能となった.

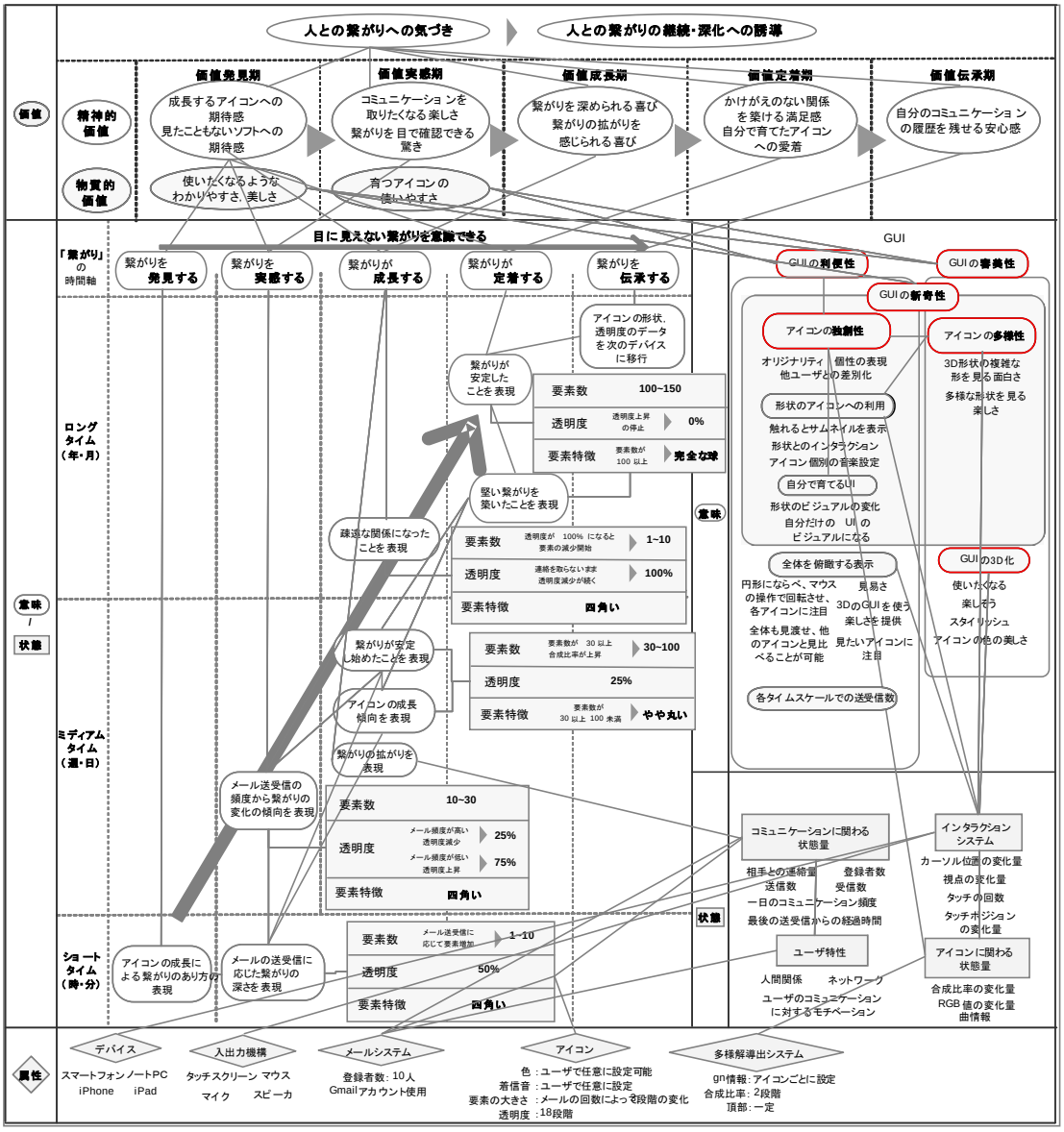


図 2-2　１次デザイン展開における要素間関係図

2.2 1 次デザイン解

2.2.1 Eメールシステムの概要

本 E メールシステムは，繋がりの状態変化に対するユーザの気づきを促し，コミュニケーションを促進することを狙いとしている．具体的には，E メールの送受信頻度とその時間経過を登録した相手を表すアイコンのビジュアルの変化によって可視化することで，繋がりの状態変化をユーザに示唆する．アイコンとは，図 2-3 に示す操作画面における各形状のことであり，それぞれが連絡先の相手を表している．次に基本仕様を述べる．

本 E メールソフトの基本仕様を次に示す．

使用機種：タブレット型ＰＣ

使用環境：Windows7

使用言語：Processing

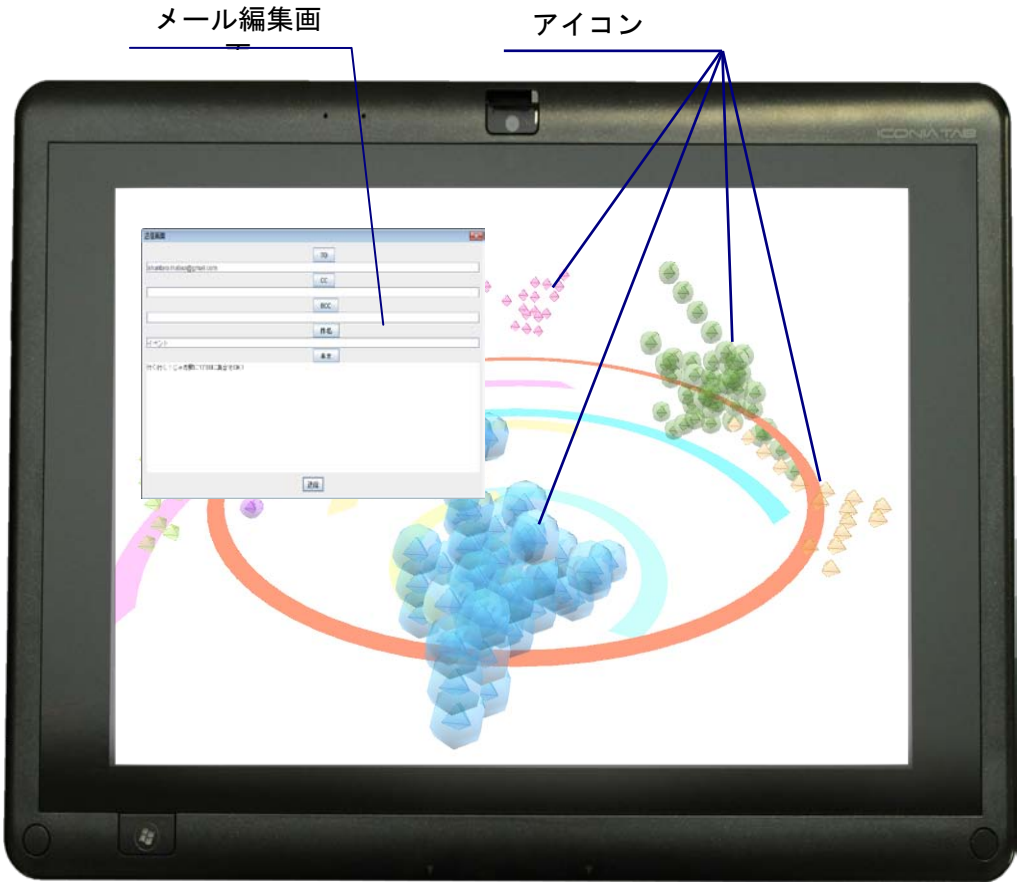


図 2-3　E メールシステムの操作画面

また，必要とされる基本性能とシステムとして，E メールの送受信システム，送受信頻度とその時間経過の表示システム（2.3 で詳述），タッチ操作による直感的操作システム，およびセルラオートマタによるアイコン制御システムが挙げられる．

2.2.2 GUI

ここでは，導出された GUI の基本仕様について述べる．

図 2-4 に示すように，1 次デザイン展開の結果，GUI には，その利便性，審美性が求められ，特に，「アイコンの独創性」，「アイコンの多様性」などによる「GUI の新奇性」が必要と考えた．そして，その意味からも，「使いたくなる」，「楽しそう」などを具現化するために，「GUI の 3D 化」が有効であると判断した．

以上の理由から，本 GUI には，アイコンの位置が 3 次元的に回転する，奥行き感のある 3D 表現とすることにした．

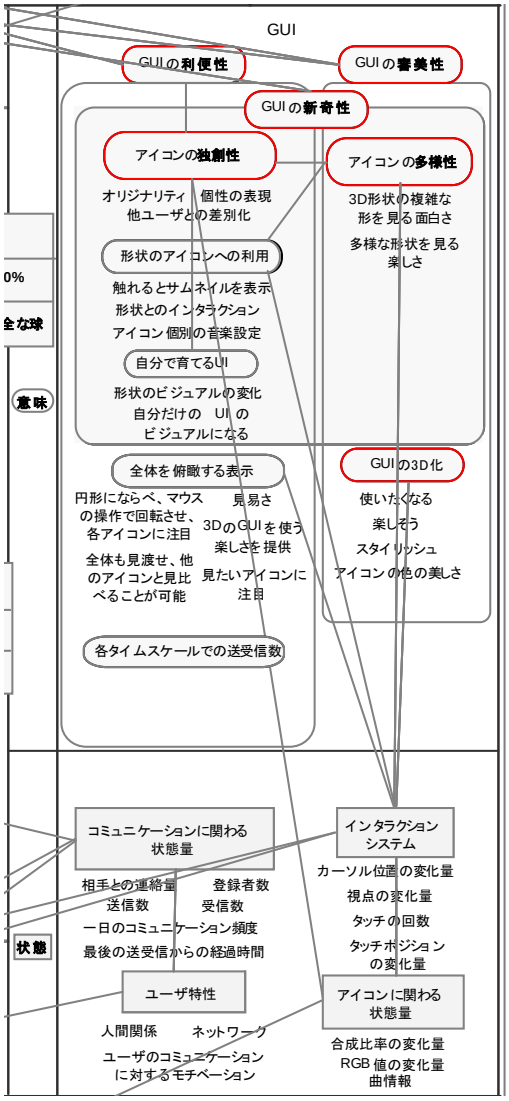


図 2-2　GUI に関する要素間関係図

2.2.3 アイコンによる6つの機能

本 E メールシステムにおいて最も重要な E メールの送受信頻度とその時間経過を可視化する機能について説明する．アイコンは，アイコンを構成する「要素数」，アイコンの「透明度」および「要素の形状特徴」の 3 つのパラメータを有する．この 3 つのパラメータを図 2-5 から図 2-10 を用いて示す 6 つの機能により変化させることで，E メールの送受信頻度とその時間経過を可視化する．6 つの機能の詳細について以下に説明する．

機能 1：人との繋がりの深さを示唆するアイコンの拡大

E メールの送信または受信を行うと，アイコンの要素数が最大要素数（150 個）以下の場合，該当するアイコンの要素数が増加する．

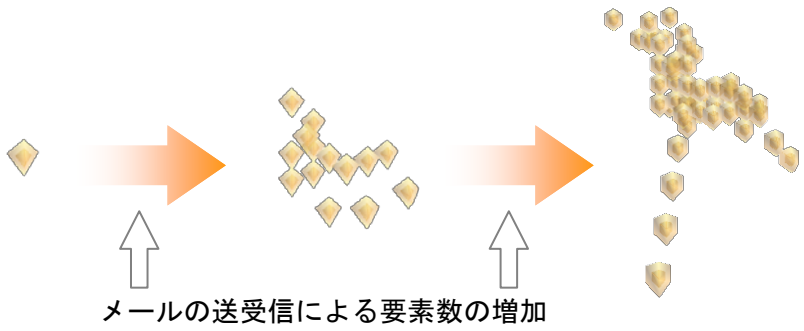


図 2-5　機能 1：人との繋がりの深さを示唆するアイコンの拡大

機能 2：人との繋がりの深さの変化を示唆する透明度変化

アイコンの透明度は 0 から 255 のレベルで設定され，0 に近づくほど透明なアイコンとなり，逆に 255 に近づくほど不透明なアイコンとなる．1 日の E メール送受信数が 10 通以上でかつ透明度が 240 以下のとき透明度が 15 レベル分増加する．また，1 日のメール送受信数が 10 通未満かつ透明度が 15 以上のとき透明度が 15 レベル分減少する．

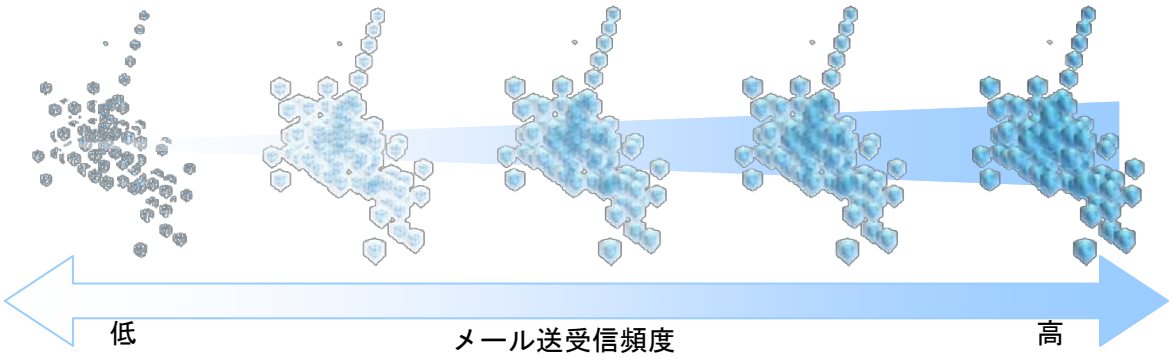


図 2-6　機能 2：人との繋がりの深さの変化を示唆する透明度変化

機能 3：人との繋がりの薄さを示唆するアイコンの縮小

アイコンの透明度が 0 の状態で 24 時間経過するごとに，最も新しい世代の要素から順に削除され，要素数が減少する．要素数が 1 個の初期状態になると，アイコンの透明度が 0 の場合でも要素は削除されない．

図 2-7　機能 3：繋がりの薄さを示唆するアイコンの縮小

機能 4：安定した人との繋がりを示唆するアイコンの成長傾向の変化

要素の発生方向を制御するために設定した形状操作パラメータの変化（0 から 1 まで）により，アイコンの成長傾向を変化させる．具体的には，要素数が 20 個未満の状態のときには，形状操作パラメータは低い数値に設定されており，細い棒状のアイコンに成長する傾向が強い．要素数が 20 個以上の状態のときには，形状操作パラメータが高い数値に変更され，大きく塊状の形状に成長する傾向が強くなる．

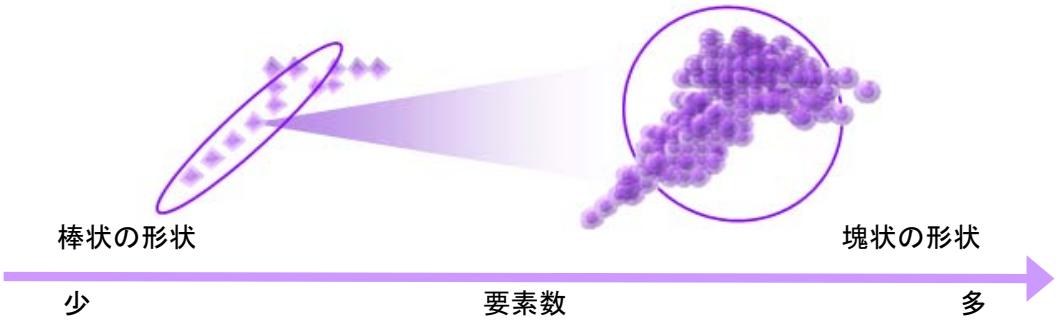


図 2-8　機能 4：人との安定した繋がりを示唆するアイコンの成長傾向の変化

機能 5：人との繋がりの成長段階を示唆する形状特徴の変化

アイコンは，時間が経過するごとに，要素数と透明度の値に応じて，球状の要素の解像度と要素の大きさを変化させる．要素数が少なく，透明度が 0 に近いほど小さく角ばった荒い球状の要素となり，要素数が多く，透明度が 255 に近いほど大きく滑らかな球状の要素となる．

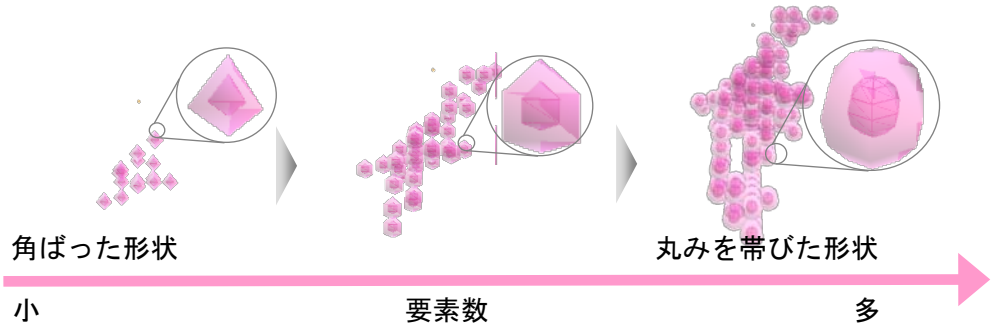


図 2-9　機能 5：人との繋がりの成長段階を示唆する形状特徴の変化

機能 6：人との繋がりの広さを示唆する新規アイコンの生成

新規に登録する相手の名前，メールアドレスを記載した txt ファイルを読み込むことで，新規アイコンを生成する．新規アイコンが生成されると，円状に配置されたアイコンの輪が広がり，新規アイコンが輪に加わるように描画される．

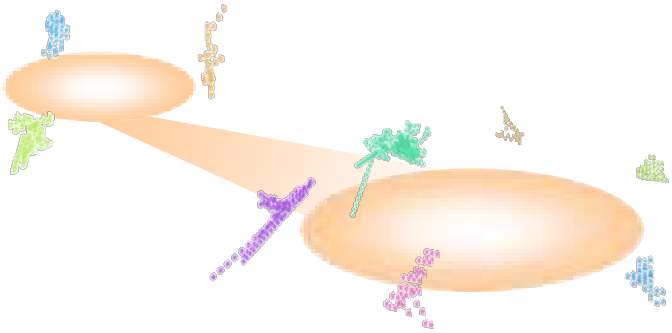


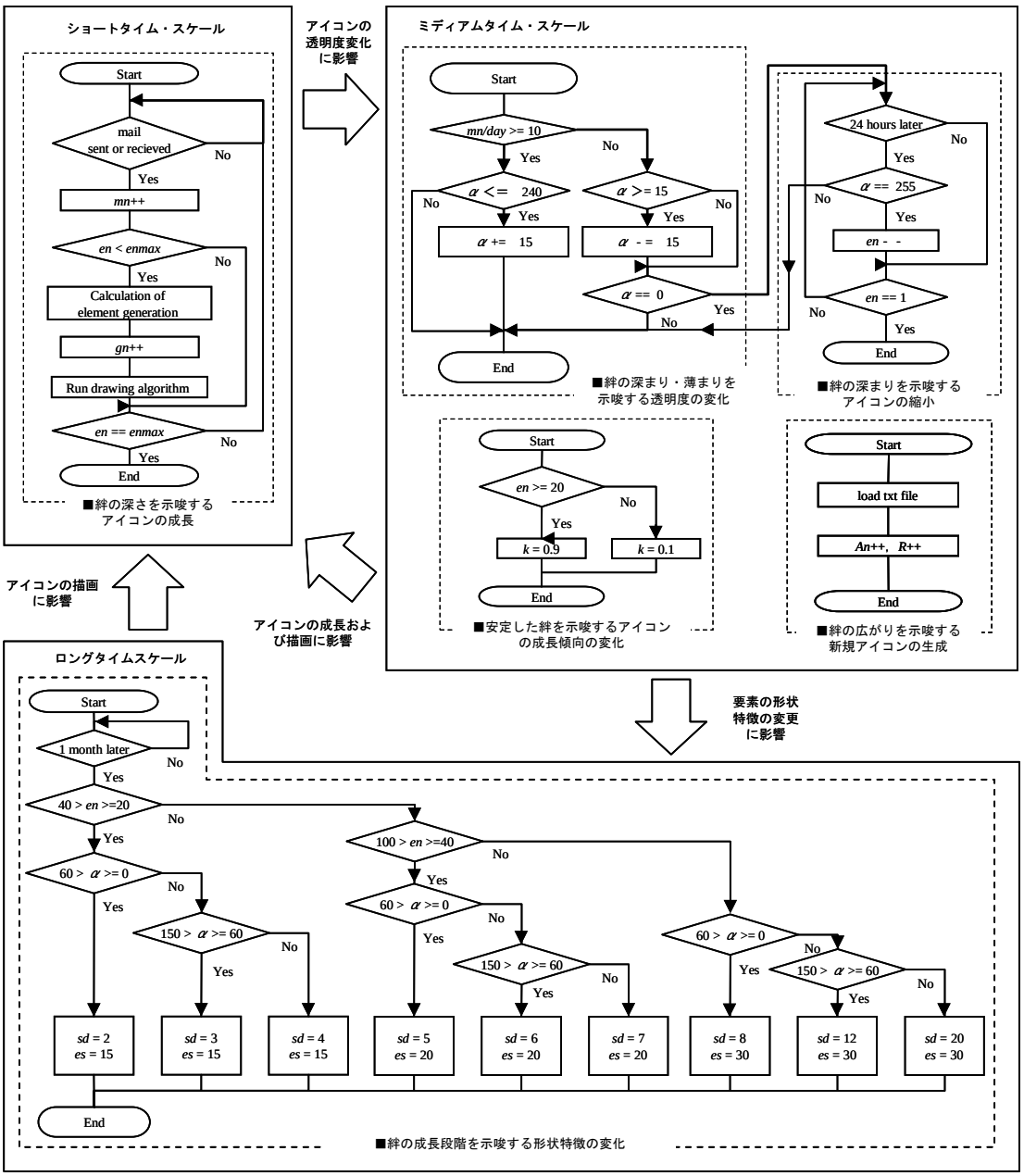
図 2-10　機能 6：人との繋がりの広さを示唆する新規アイコンの生成

なお，以上に示した 6 つの機能に関して，その具体的な機能と効果について，表 2-1 にまとめた．これらの機能により，アイコンの大きさ，形状，透明度の状態・状態の変化とアイコンの間（送受信者間）の比較から人との繋がりの状態・状態の変化に気づきを与えることを狙いとした．

また，これらの 6 つの機能を，ショートタイム・スケール，ミディアムタイム・スケールおよびロングタイム・スケールごとにまとめ，それらの関係性を示したシステムフローを図 2-11 に示す．

表 2-1　6 つの機能における機能と効果

機能名称	内容			
	属性	状態	機能内容	効果
機能 1：人との繋がりの 深さを示唆する アイコンの拡大	アイコンの 大きさ （要素数）	状態	送受信数n回 ⇒n世代分の要素数の追加 （1≦要素数≦150）	人との繋がりが深化する様子を示唆し， コミュニケーションをとる楽しさを提供する．
機能 2：人との繋がりの 深さの変化を示唆する 透明度変化	アイコンの 透明度	状態変化	10通/24h以上で透明度240以下 ⇒透明度+15 10通/24h未満で透明度15以上 ⇒透明度−15 （0（透明）≦透明度≦255（不透明））	人との繋がりの深さの変化を認識でき， 人との繋がりを深める意識を促す．
機能 3：人との繋がりの 薄さを示唆する アイコンの縮小	アイコンの 大きさ （要素数）	状態	透明度0の時，24時間毎に要素数1世代分減少 （要素数＝1の時要素数は減らない）	アイコンの透明度が0になると，要素数が 減少し人との繋がりの薄さを気付かせる．
機能 4：安定した人との 繋がりを示唆する アイコンの成長傾向の変化	アイコンの 形状特徴	状態の変化	要素数20未満 ⇒形状操作パラメータは低い 要素数20以上 ⇒形状操作パラメータは高い （0≦形状操作パラメータ≦1）	アイコンが一定の大きさを超えると 塊状になりやすくなり，人との繋がりが 安定したような印象を与える．
機能 5：人との繋がりの 深さの段階を示唆する 形状特徴の変化	要素の形 状特徴	状態の変化	要素数が少なく，透明度が0に近い時 ⇒角ばった形状 要素数が多く，透明度が255に近い時 ⇒丸みを帯びた形状	安定した人との繋がりがさらに成長してい く状態を段階的に認識でき，深い人との繋 がりを築く満足感を提供する．
機能 6：人との繋がりの 広さを示唆する 新規アイコンの生成	アイコンの 追加	状態	アイコン+*n* 個	アイコンの輪の広さによって， 人間関係を拡げる楽しみを提供する．



en	:要素数	*R*	:アイコンが配置される直径の大きさ	*α*	:透明度	*gnmax*	:最大世代数
enmax	:最大要素数	*An*	:アイコン数	*k*	:形状操作パラメータ	*mn*	:メール送受信数
gn	:世代数	*es*	:要素のサイズ	*sd*	:要素の球状ポリゴンの解像度	*mn/day*	:一日のメール送受信数

図 2-11　各機能のシステムフロー図

2.3 1次プロトタイプの評価

2.3.1 官能評価実験の概要

本Eメールシステムにおける繋がりを示唆するアイコンの6つ機能の表現の有効性と本Eメールシステムにおける各種の評価を確認するため，本Eメールシステムのプロトタイプを用いて，官能評価実験を実施した[7,8].

実験の対象は, 20歳代の学生34名とした. また, この集団における男女比は1：1であった. 本実験に用いた質問内容を表2-2に示す. 質問に対する評価項目は, 質問1 ～ 6においては，1.全く思わない　2.思わない　3.どちらとも言えない　4.思う　5.とても思う，の5つとした[9,10,11]. また，質問7～11における評価項目は，1.全く評価できない　2.評価できない　3.どちらでもない　4.評価できる　5.とても評価できるの5つとし，5段階絶対評価によるSD法を採用した．また，本Eメールシステムに関する質問の前に，PC，タブレットおよびスマートフォンの日常的な使用の有無を確認した.

表 2-2 質問項目

No.	質問内容
質問 1	人との繋がりをアイコンの変化によって表現できていると思いますか？
質問 2	人との繋がりを透明度によって表現できていると思いますか？
質問 3	人との繋がりをアイコンの縮小によって表現できていると思いますか？
質問 4	アイコンが塊状になることで安定した人との繋がりを表現できていると思いますか？
質問 5	人との繋がりの成長段階を, 要素の形状変化によって表現できていると思いますか？
質問 6	人との繋がりの広がりを, 新規アイコンの生成によって表現できていると思いますか？
質問 7	企画性：コンセプトや独創性など
質問 8	機能性：人との繋がりの気づき
質問 9	操作性：使いやすさ(ここでの「使いやすさ」は単にoperationのみならず, 視認性などを含む広義の意味でのusabilityを指す)
質問 10	意匠性：グラフィックの良さ
質問 11	全体評価
質問 12	プロトタイプ全体に対するコメント

2.3.2 実験結果の解析とその考察

官能評価実験の結果から，評価点において男女間の評価点に差があるかの確認を行ったが，男女間で評価点における有意差は確認されなかった．また，タブレット，E メール，およびスマートフォンの日常的な使用の有無に関しては，被験者全員が PC と E メールを日常的に使用していることがわかった．一方で，タブレットを日常的に使用してる被験者は 1 名のみであることがわかった．図 2-12 に評価項目および実験結果である各評価項目の評価点平均を示す．E メールシステム全体に対する評価の平均点は 3.79 であり，またすべての評価項目において平均点が 3.5 以上であった．この結果より，E メールシステム全体としてはその有効性に対して一定の評価が示されたと考えられる．特に，企画性は平均点が 4.09 と最も高く，以下のようなコメントもあり，その独自性が高く評価されている．

・新しいコミュニケーションの形態として面白いと思う．

・アイコンで画面をいっぱいにしたくなる．

一方で全体コメントにおいては，以下のような改善点を指摘するコメントもあった．

・アイコンの形態（形状，色彩）だけでは，アイコンが有する送受信先を識別することが難しく，送受信先の名前が表示されることが望ましい．

・６つの機能によるアイコンの変化は，直感的認識には一定以上の評価が得られているものの，定量的な連絡履歴の履歴が表示されることが望ましい．

などの，今後の E メールシステムの開発において有益な意見を得ることができた．特に，評価点の低い項目に着目すると，操作性に対する評価平均点は 3.53 であり，最も低い値をとっている．そのため，操作性に関しては，さらなる改善が望まれる．

この結果を受け，操作性に対する評価点を目的変数，各機能に対する評価点を説明変数として，重回帰分析を行い，操作性の向上に有効な機能の抽出を行った．重回帰分析によって得られた重回帰式を式(2-1)に示す．

$$Q_U = 0.059Q_1 - 0.255Q_2 - 0.158Q_3 + 0.331Q_4 + 0.446Q_5 + 0.351Q_6 \qquad (2\text{-}1)$$

式(2-1)の標準偏回帰係数より，操作性の評価点には機能 4，機能 5，および機能 6 が寄与していると考えられる．したがって，機能 4，機能 5 および機能 6 の向上によって操作性の評価の向上すると考えられる．また，これらの 3 つの機能のうち，機能 4 は，棒状などの不安定さを想起させる形状から塊状などの安定した印象を想起させる形状へと変化させ，繋がりの安定を示唆する機能である．そのため，多様解導出システムにおける適切なシステムパラメータの設定を行い，今後，機能 4 のさらなる向上が望まれる．

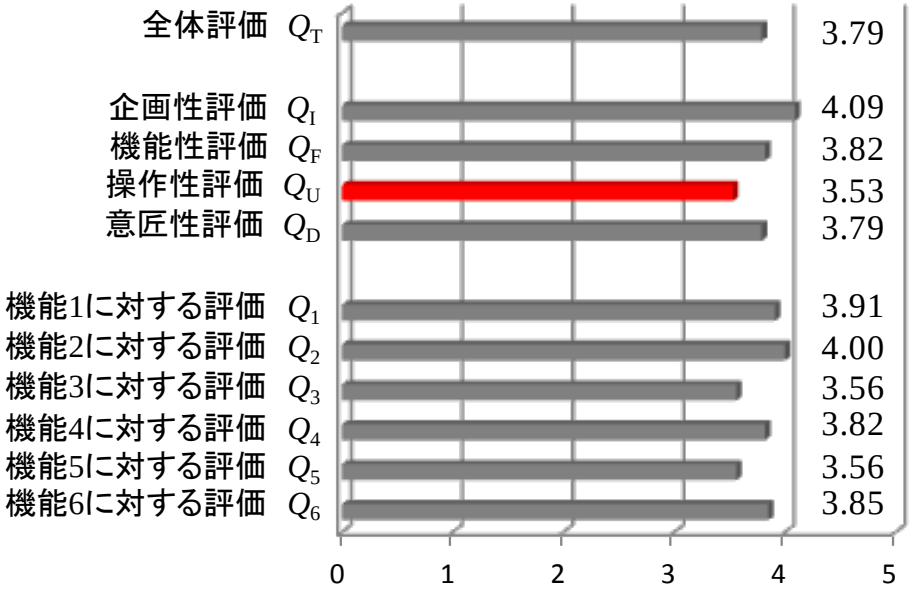


図 2-12 各評価項目の評価点平均

2.3.3 1 次プロトタイプ全体の考察

ここでは，官能評価実験の結果を踏まえて，本 E メールシステムによる繋がりの継続・深化の実現の可能性を考察する．図 2-13 に本 E メールシステムによる「繋がり」の深化の過程を示す．この過程に沿って以下のように考察した．

価値発見期では，ユーザに本 E メールシステムへの期待を抱かせる必要がある． E メールの送受信頻度を成長するアイコンで可視化し，「繋がり」への気づきを表現するという新しい「繋がり」のあり方を，店頭や web 上でのデモムービーなどで見せることで，ユーザの期待感を高めることが可能になると考えられる．実際に，アンケート結果において企画性に関して高い評価を受けていることと，「新しいコミュニケーションの形態として面白いと思う.」というコメントを受けており，本 E メールシステムへの期待をユーザに抱かせることは可能であると考えられる．

価値実感期では，ユーザが実際に製品を使用し，繋がりを実感する必要がある．この期においては，ユーザはメールの送受信頻度に応じたアイコンの成長を経験することで，成長するアイコンに対する驚きや，「繋がり」が深まっていくように感じる楽しさを実感することができると考えられる．アンケートにおいても「使ってしてわくわくするし楽しい」といったコメントを受けており，ユーザにもっと使い続けてみたいと感じさせることができると考えられる．

価値成長期では，ユーザに繋がりの深まりを感じてもらう必要がある．この期においては，コミュニケーションを頻繁にとり大きくなったアイコンや，逆に透明になり小さくなり始めたアイコンなどが混在していると考えられる．そのため，様々な相手との「繋がり」の深まり，薄まりをリアルタイムに実感し，より繋がりを深めたいという思いや繋がりを再び深めたいといった思いをユーザに感じさせることが可能になると考えられる．アンケート結果からも「人間関係の疎密が一目でわかる」「友人と繋がっていない！とあせるかもしれない」といったコメントを受けている．以上より，ユーザに，繋がりの深まり感じさせ，「繋がりをもっと深めたい，継続したい」という意欲を促すことができると考えられる．

価値定着期では，E メールシステムを使い込み，安定した繋がりを感じるこ

とが必要となる．この期においては，要素数が増え大きく安定した印象を与えるアイコンが増えている．そのため，ユーザは家族や友達と安定した「繋がり」を築いた達成感や満足感を感じることができると考えられる．アンケート結果からも，「要素の数がその人との思い出の数だと感じることができる」というコメントを受けており，大きく安定したアイコンから思い入れを感じることができることが示唆されている．したがって，本 E メールシステムにより，自身の築いた繋がりに対し愛着を感じ継続していくことを促すことができると考えられる．

最後に，価値伝承期では，思い入れを強めた繋がりを継続できる安心感を抱かせることが必要である．この期においては，成長したアイコンの情報を次のデバイスに移行することで，自身の積み重ねた人間関係を継続していくことができる安心感を得られると考えられる．

以上の考察から， 5 つの期間に分けた繋がりの継続・深化の過程において，本 E メールシステムは求められるニーズ，価値を対応することが可能であり，繋がりの継続・深化を実現できる可能性が示された．

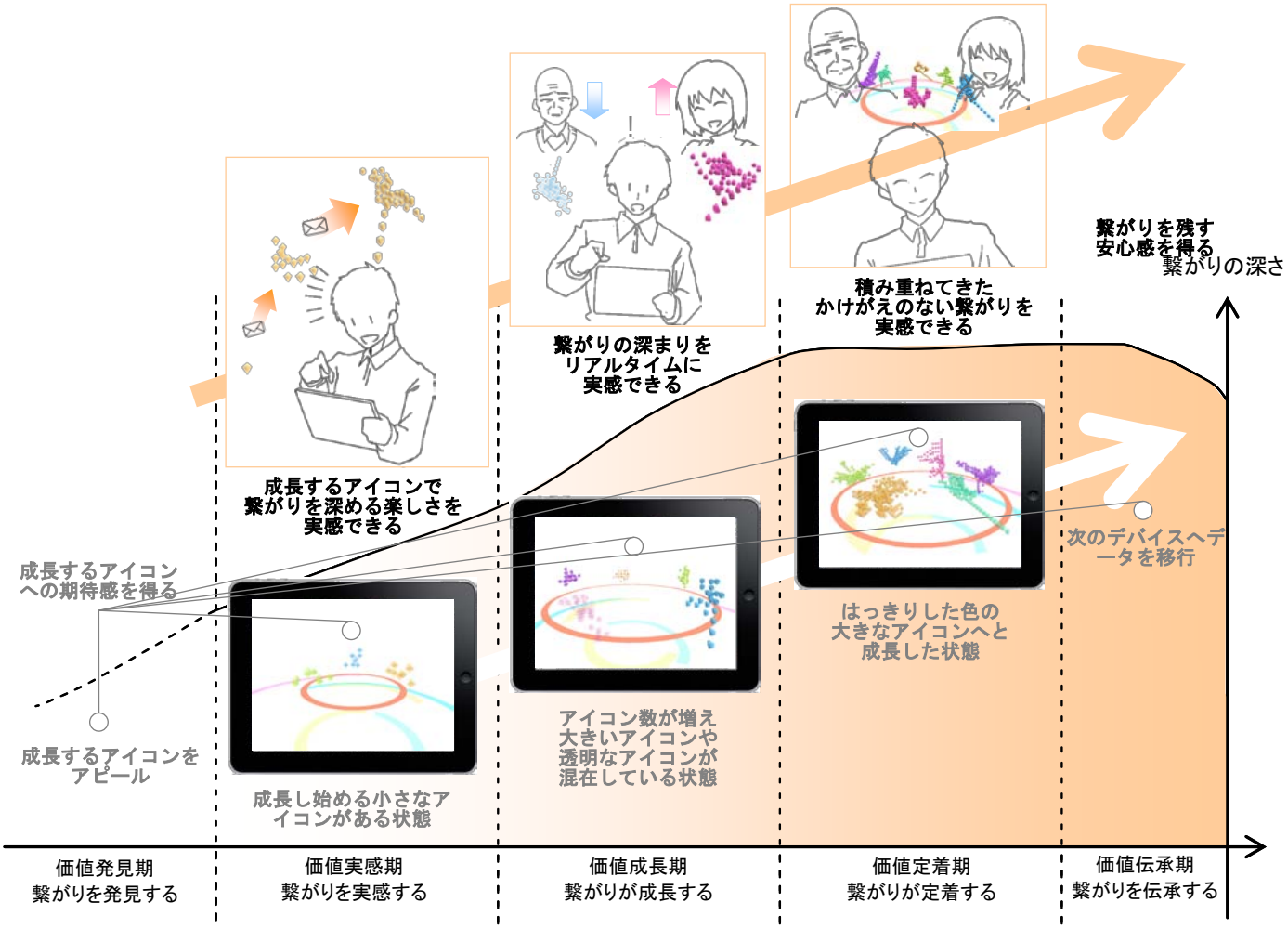


図 2-13 導出した E メールシステムによる「繋がり」の深化の過程

第３章

２次デザイン展開：
送受信数履歴の表示

3.1　2 次デザイン展開

3.1.1 デザイン要素の抽出

第 2 章で示した 1 次プロトタイプによる官能評価実験（SD 法による 5 段階絶対評価）の結果，企画性，機能性，操作性，意匠性，全体評価のすべての項目において，3.5 点以上という一定以上の評価を得ている．しかしながら，その項目の中でも，特に操作性は 3.53 点と比較的低く，改良の余地を残しているといえる．そのため，ここでは，操作性の改良を狙いとして，さらなる 2 次デザイン展開を行った．

操作性の改良の視点としては，1 次プロトタイプによる官能評価実験の際に得られたコメントから，以下の 2 点に注目した．

- アイコンの形態（形状，色彩）だけでは，アイコンが有する送受信先を識別することが難しく，送受信先の名前が表示されることが望ましい．
- ６つの機能によるアイコンの変化は，直感的認識には一定以上の評価が得られているものの，定量的な連絡履歴の履歴が表示されることが望ましい．

以上の 2 つの視点から，デザイン要素の抽出を行った．その際，グラフィック・ユーザ・インターフェイス（GUI）に注目し，図 3-1 中の右側の GUI に示すように，意味要素として，「送受信先の識別」，「送受信先との連絡履歴の理解」が抽出され，それぞれに対応する状態要素として，「送受信先の表示」，「送受信先の連絡履歴の表示」が得られ，新たな意味と状態の要素を導出した．

3.1.2 デザイン要素の構造化と機能の導出

ここでは，3.1.1 節で得られた意味要素と状態要素に基づいて具体的な構想をシステムとして構築するため，デザイン要素の構造化を行った．その結果である要素間関係図を図 3-1 に示す.

まず，意味要素「送受信先の識別」と状態要素「送受信先の表示」の関係性から属性要素「送受信先の名前の表示追加」という機能が抽出された．その際，すでに設定されていた意味要素「GUI の審美性」や「GUI のビジュアル」との関係性から，通常の不使用時には送受信先の名前は表示されず，必要な使用時のみ送受信先の名前が表示されることとした．その結果，カーソルを送受信先のアイコンに合わせた際に送受信先の名前が表示される機能とし，GUI の煩雑さを避け，審美性を確保することを考慮した．また，名前の表示色は，送受信先ごとに色分けしてあるアイコンの色に合わせることで，送受信先の識別を容易にした.

次に，意味要素「送受信先との連絡履歴の理解」と状態要素「送受信先との連絡履歴の表示」の関係性から属性要素「１週間，１か月間，１年間の送受信数履歴の表示追加」という機能が抽出された．その際，すでに設定されていた意味要素「各タイムスケールでの送受信数」との関係性から，１週間，１か月間，および１年間という 3 つのタイムスケールでの送受信数の履歴を示すことが有効であると考えた．これにより，単に送受信頻度だけではなく，その頻度の変化を実感できることを考慮した．また，その履歴の表示は，名前の表示色と同様に，カーソルを送受信先のアイコンに合わせた際にのみ表示される機能とし，表示色についても，アイコンの色に合わせることとした.

「送受信先の連絡履歴の表示」を 2 次プロトタイプに加えるために，データを蓄積・格納するシステムが必要であると考えた．しかしながら，1 次プロトタイプにおいて作成した POP サーバーを用いたメールの受信方式では履歴の保存が難しいことから，2 次デザイン展開では，IMAP サーバーを用いた受信方式に変更した．この変更により，「送受信先の連絡履歴の表示」に用いる定量的なデータを蓄積・格納することを可能にした.

以上に示したように，1 次プロトタイプの評価結果を受け，操作性の向上を主

な狙いとし，2 次デザイン展開を行った．その結果，E メールシステムの GUI において，「送受信先の名前の表示」および「１週間，１か月間，１年間の送受信数履歴の表示」とい 2 つの機能を追加することとなった．

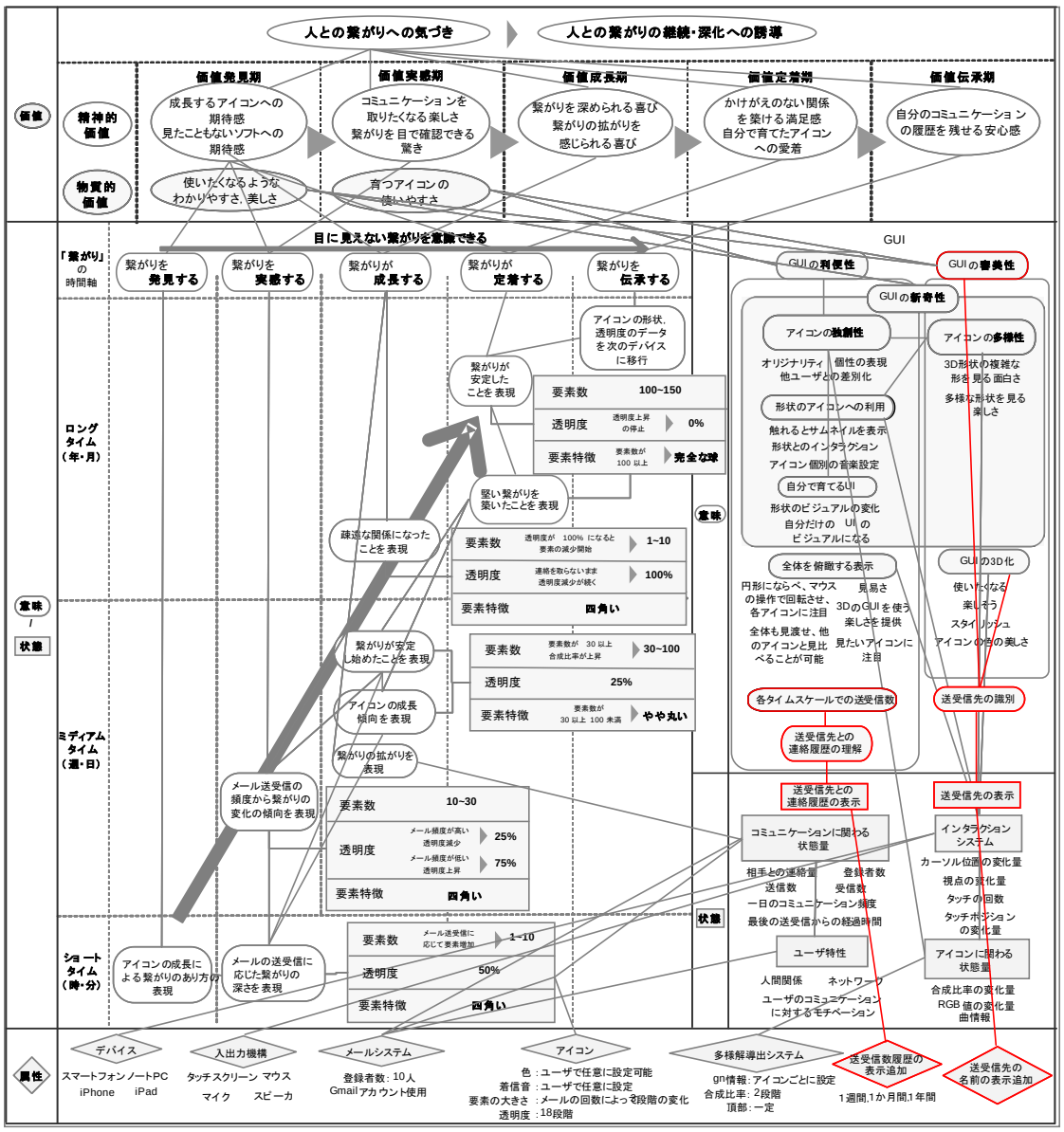


図 3-1　2 次デザイン展開における要素間関係図

3.2 2 次デザイン解

3.2.1 送受信先の名前表示機能

3.1 節で示したように，2 次デザイン展開においては 2 つの機能をデザイン解として追加することにした．ここでは，その 1 つ，「送受信先の名前の表示」について，その基本仕様を以下に説明する（図 3-2 参照）．

・名前の表示のタイミング：通常の不使用時には送受信先の名前は表示されず，必要な使用時のみ送受信の名前が表示される．そのため，カーソルを送受信先のアイコンに合わせた際にのみ送受信先の名前が表示される機能とする．（審美性の考慮のため．）

・名前の表示色：送受信先ごとに色分けしてあるアイコンの色に合わせる．（審美性の考慮と送受信先の識別を容易にするため．）

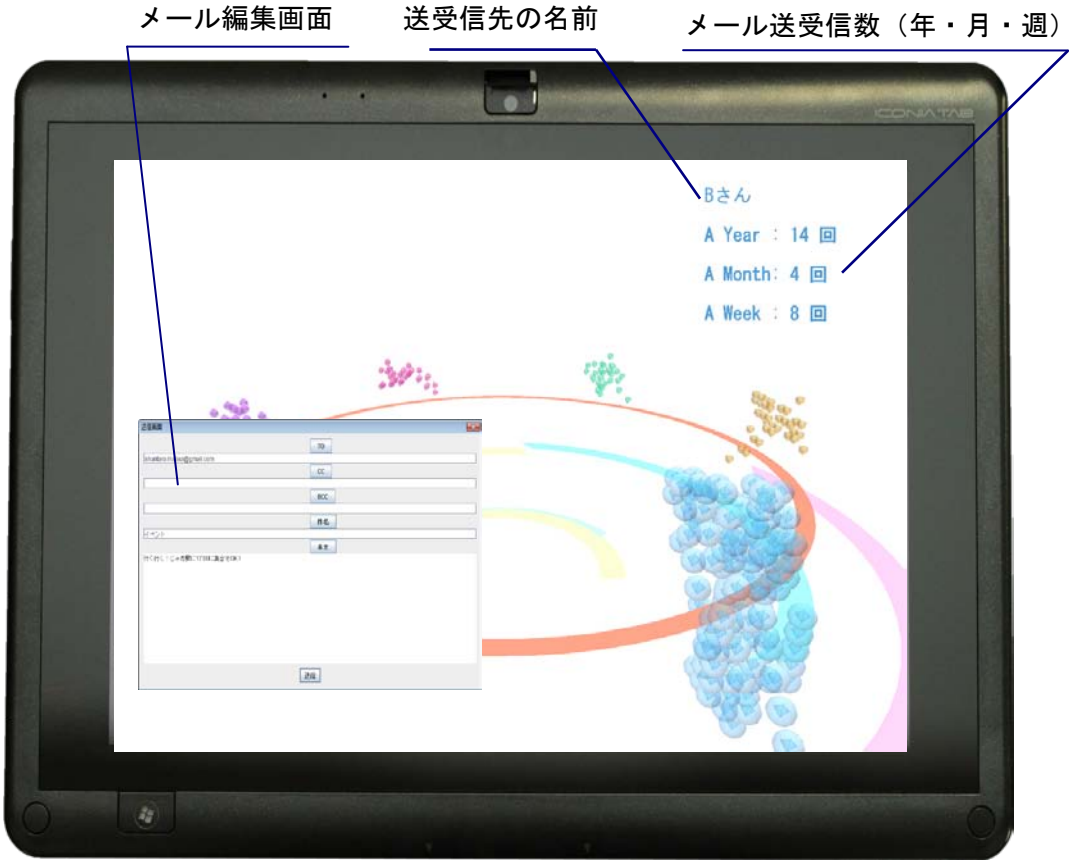


図 3-2　2 次デザイン解の操作画面

3.2.2 送受信数の履歴表示機能

ここでは，2 次デザイン展開において得られたもう 1 つの機能「1 週間，1 か月間，1 年間の送受信数履歴の表示」について，以下にその基本仕様を説明する（図 3-2 参照）.

・送受信数履歴の表示内容：1 週間，1 か月間，および 1 年間という 3 つのタイムスケールでの送受信先との送受信数の履歴を示す.（単に送受信頻度だけではなく，その頻度の変化を実感できることを考慮.）なお，各タイムスケールの計算は，便宜上，以下による.

1 週間　　：使用時点から 168 時間（＝24 時間×7 日）前から使用時点まで

1 か月間　：使用時点から 720 時間（＝24 時間×30 日）前から使用時点まで

1 年間　　：使用時点から 8,760 時間（＝24 時間×365 日）前から使用時点まで

・送受信数履歴の表示のタイミング：送受信先の名前表示と同様に，通常の不使用時には表示されず，必要な使用時のみ送受信の名前が表示される．そのため，カーソルを送受信先のアイコンに合わせた際にのみ送受信先の名前が表示される機能とする.（審美性の考慮のため.）

・送受信数履歴の表示色：送受信先の名前表示と同様に，送受信先ごとに色分けしてあるアイコンの色に合わせる.（審美性の考慮と送受信先の識別を容易にするため.）

3.3 2次プロトタイプの評価

3.3.1 官能評価実験の概要

3.1節および3.2節において示したように，2次デザイン展開においては，主として操作性の向上を狙いとし，「送受信先の名前の表示機能」および「送受信先との送受信数履歴の表示機能」という2つの機能を追加した．ここでは，それらの導出された機能の有効性を確認するために，制作した2次プロトタイプを用いて，官能評価実験を実施した．

実験の対象は，20歳代の学生12名とした．本実験に用いた質問内容を表3-1に示す．質問に対する評価項目は，1.全く思わない　2.思わない　3.どちらとも言えない　4.思う　5.とても思う，の5つとし，5段階絶対評価によるSD法を採用した．

表 3-1 質問項目

No.	質問内容
質問 1	企画性：コンセプトや独創性など
質問 2	機能性：人との繋がりの気づき
質問 3	操作性：使いやすさ（ここでの「使いやすさ」は単にoperationのみならず，視認性などを含む広義の意味でのusabilityを指す）
質問 4	意匠性：グラフィックの良さ
質問 5	全体評価
質問 6	プロトタイプ全体に対するコメント

3.3.2 実験結果の解析とその考察

官能評価実験の結果（平均点）を，1 次プロトタイプに対する評価結果とともに，図 3-3 に示す．この図が示すように，2 次プロトタイプが狙いとしていた操作性については，1 次プロトタイプの評価点が 3.53 点であるのに対して，4.33 点と＋0.80 点の大幅な向上が見受けられ，一定の効果が確認できたといえる．

また，その他の項目に目を向けると，意匠性に関しては，ほぼ同等であるものの，企画性に関しては 4.09 点から 4.50 点へと＋0.41 点，機能性に関しては 3.82 点から 4.33 点へと＋0.51 点，さらに全体評価に関しても 3.79 点から 4.08 点と＋0.29 点の向上が見受けられた．

なお，得られたコメントについても，以下のような良好な結果であった．

- これまで視認できていなかった人との繋がりの度合いを確認できて企画としてとても興味深いと思った．
- アイコンの形状変化は良いと感じました．特に 1 つひとつが小さなセルでそれにより全体が構成されている点がおもしろいと感じました．

ただし，以下のような改善の指摘も得られ，今後の参考としたい．

- カーソルを合わせた時に，時間毎にどうアイコンが変化したかが可視化されていると良いと思います．(現在のような，回数だけではなく図示する．)

以上より，２次プロトタイプに関しては，官能評価実験の結果は良好であった．これは，操作性向上を狙いとして 2 次プロトタイプに採用した「送受信先の名前の表示機能」および「送受信先との送受信数履歴の表示機能」という 2 つの機能により，企画性や機能性の向上にも貢献し，その結果，全体評価が向上したものと考える．このことは，この 2 つの機能が操作性以外にも有効であったことを示唆していると考察できる．

意匠性評価については，1 次プロトタイプからの向上がわずかであることから，今後の意匠性の向上が望まれる．

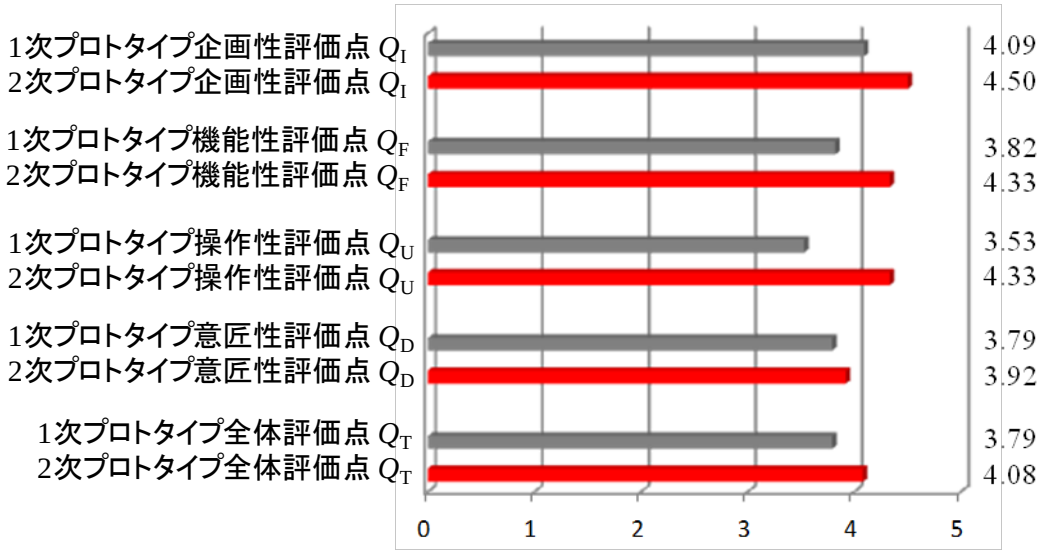


図 3-3 各評価項目の評価点平均

3.3.3 2次プロトタイプ全体の考察

最後に，官能評価実験の結果を踏まえて，2 次プロトタイプにおける可視化に関する 2 つの機能を, 1 次プロトタイプにおける可視化に関する機能との関係性から考察する．

2 次プロトタイプにて追加した 2 つの機能,「送受信先の名前の表示機能」および「送受信先との送受信数履歴の表示機能」は，1 次プロトタイプにおいて採用した 6 つの機能における認知上のあいまい性を補うべく採用した機能である．その効果は，先述した官能評価実験の結果である操作性評価の向上で認められる．

ここで, 1 次プロトタイプで採用した 6 つの機能と 2 次プロトタイプで採用した 2 つの機能の双方の役割を考える．前者の 6 つの機能は，アイコンの形状や透明度という色彩の時間的変化から人との繋がりの変化を感性的に認識するための可視化である．このタイプの可視化は，デジタル情報と異なり，不使用時においても煩わしさをさほど感じない利点がある．また，形状や色彩の変化に対する気づきは，ユーザの感性によるものであることから，その面白さから感性的な価値や意味があるといえる．

その一方，後者の 2 次プロトタイプで採用した 2 つの機能はデジタルによる可視化であり，前者のあいまい性を補うことを狙いとした理性的認識のためのものである．これにより，人との繋がりをより理性的に理解するための機能であるといえる．しかし，このタイプの可視化は，不使用時において常に表示されると，GUI としては煩わしさを否めない．そのため，本 2 次プロトタイプにおいては, 6 つの機能により感性により感じ取った気づきをより明確にするため, 機能を追加したといえる．このような感性的認識と理性的認識は補完的であるといえ，両者の採用により，2 次プロトタイプは一定以上の評価が得られ，人との繋がりへの気づきを与えるものと評価されたと考える．

第 4 章

結言

4.1 本研究の成果

本研究では，家族や知人をはじめとした，人との繋がりという精神的な豊かさを求める気運の高まりを受け，その繋がりの継続・深化に向けた E メールシステムのデザインを試みた．本研究の成果を以下に示す．

(1) 週単位，月単位，年単位などのさまざまなタイムスケールを総合的に取り扱うマルチタイムスケールの視点から 1 次デザイン展開を行い，送受信数の履歴を，形状や透明度などが変化するアイコンにより可視化することができた．

(2) 1 次デザイン展開から得られたデザイン解をもとに，実際にプログラムを作成し，E メールシステムの 1 次プロトタイプを製作した．

(3) 1 次プロトタイプを用いた官能評価実験を実施し，その評価結果から一定以上の評価を受け，本 E メールシステムの有効性が示されたものの，操作性の評価については，まだ改良の余地があることを確認した．

(4) 上記の評価結果から，2 次デザイン展開を行い，アイコン表示に，送受信先の名前と，ここ 1 年間，1 か月間，1 週間の各送受信数を追加して可視化する改良を加えた 2 次デザイン解を導出した．

(5) 2 次デザイン解に関して，プログラムを作成し，2 次プロトタイプを製作した．

(6) 2 次プロトタイプを用いた官能評価実験を実施し，その操作性の評価も向上し，本 E メールシステムの有効性が示された．

以上の成果より，E メールの送受信数を可視化することで人との繋がりの状況に対する気づきをユーザに与え，その継続・深化に向けたコミュニケーションを促すべく，E メールシステムをデザインした．これにより，今日において社会的な課題である，人との繋がりの不足に対して，それを充足する一助とすることができるものと考える．

4.2 今後の課題

今後の課題としては以下の項目が挙げられる．

(1) 本 E メールシステムを用いて，長期間の使用実験を行い，人との繋がりに対しての有効性を確認する．

(2) 機能 4 は，棒状などの不安定さを想起させる形状から塊状などの安定した印象を想起させる形状へと変化させ，繋がりの安定を示唆する機能であり，今後，多様解導出システムにおける適切なシステムパラメータの設定による，さらなる機能の向上が望まれる．

(3) カーソルを合わせた時に，タイムスケールごとのアイコンの変化を図示し，可視化する．

(4) 感性的認知と理性的認知のバランスを再考することで，機能やプログラムの最適化を行い，意匠性を含むさらなる評価の向上を図る．

以上の項目に対して研究を進め，より人との繋がりに貢献できる E メールシステムへと改良していく．

謝　辞

本研究は，慶應義塾大学大学院システムデザイン・マネジメント研究科，小木哲朗教授のご指導のもとで行われたものです．本研究を遂行するにあたり，多大なご指導，ご鞭撻をいただきました小木哲朗教授に心よりお礼申し上げます．

また，本研究の関わる全ての事柄に対して，副査としてさまざまな視点からご指導いただきました西村秀和教授にも感謝申し上げたいと存じます．

さらに，研究面でのご指導を頂戴いたしました小木哲朗研究室の立山義祐先生をはじめとして，多くの先輩方，同期の皆様には心よりお礼申し上げます．

最後に，学生生活において私を支えてくれた家族，友人に，心より感謝の意を表したく思います．

2014 年 1 月 22 日

参考文献

[1] Wolfram, S. : Theory and Application of Cellular Automata, World Scientific, 1996

[2] 都倉信樹：オートマトンと形式言語, 昭晃堂, 1995

[3] 加藤恭義 他：セルオートマトン法, 森北出版, 1998

[4] Kei Matsuoka, et.al, Email System "KIZUNA Visualizer" Designed by Using Multispace Design Method, Proc, UMTIK 2012 International Conference on Machine Design and Production, 2012-6

[5] Kei Matsuoka, et.al, Timeaxis Design of an Email System for Sustaining and Deepening "KIZUNA", Proc, International Conference on Advanced Collaborative Networks, Systems and Applications, 2012-6

[6] Kei Matsuoka, et.al, Timeaxis Design of Email System "KIZUNA Visualizer" by Using Multispace Method, Proc., 5th International Congress of International Association of Societies of Design Research, 2013-8

[7] 石村卓夫：SPSS による多変量データ解析の手順，東京図書，2005

[8] 高田洋 : 廣瀬穀士，村瀬洋一，SPSS による多変量解析，オーム社，2007

[9] 田部井明美：SPSS 完全活用今日文銭構造分析によるアンケート処理, 東京図書,2011

[10] 小島隆矢：Excel で学ぶ共分散構造とグラフィカルモデル，オーム社, 2003

[11] 縄田和満：Excel 統計解析ボックスによるデータ解析，朝倉書店，2001

Appendix

A.1 1 次プロトタイプ ソースコード

```
//////////////////
/////メイン関数/////
//////////////////

import com.sun.opengl.util.*;
import javax.media.opengl.*;
import processing.opengl.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;
import java.io.File.*;

int ScrSizeX = 1024;                    //ウィンドウサイズ(x 軸)
int ScrSizeY = 768;                     //ウィンドウサイズ(y 軸)
int F_num = 6;                          //アイコンの数(最初に複数個表示させたければ複数個を代入)
int sequencer = 0;                      //画面の切り替え変数
int P_NUM_S = 0 ;                       //送信した相手を確認するための変数
int ER_switch = 0;                      //受信確認時, 受信メールがあった場合、形状生成に移るための変数
int ES_switch = 0;                      //送信確認時, 形状生成に移るための変数
int New_switch = 0;                     //
int View_chara = 0;                     //
color View_color ;                      //プロフィールや
ArrayList<Emergent> Chara_list = new ArrayList<Emergent>(); //F_num の数だけのアイコンのオブジェクトを格納する配列
ArrayList<Integer> F_ELM_list = new ArrayList<Integer>(); //各キャラクタの要素数を格納する配列
ArrayList<Integer> F_clearness_list = new ArrayList<Integer>(); //各キャラクタの透明度を入れる配列
ArrayList<Integer> mail_frequency_list = new ArrayList<Integer>(); //一定期間内のメール送受信回数を入れておく配列
ArrayList<Integer> P_NUM_R_list = new ArrayList<Integer>(); //受信確認時に登録された相手から何通メールが来ていたか入れる配列
ArrayList<Integer> RM_switch_list = new ArrayList<Integer>(); //どのキャラクタが受信しているかを判定する数値を格納する配列

//オブジェクトの名称設定
GraphicInterface_E gie;
back_ground bg;
RM_view rm;
profile_view pr;
```

```
void setup() {
   size(ScrSizeX, ScrSizeY, OPENGL); //スクリーンサイズの定義
   colorMode(RGB, 255);                    //カラーモードの定義
   hint(ENABLE_OPENGL_4X_SMOOTH);      //4 倍速アンチエイリアスの定義
   frameRate(60);                    //描画フレームレート

   //オブジェクトのインスタンス化
   gie = new GraphicInterface_E();
   bg = new back_ground();
   rm = new RM_view();
   pr = new profile_view();

   //背景のセットアップ
   bg.set_up();                              //内容は Background のタブ参照

   //受信メール数の初期設定(0 通にしておく)
   for (int i=0; i<F_num;i++) {
      P_NUM_R_list.add(i, 0);
   }
   //透明度初期設定(本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませること
ことになる)
   for (int i=0; i < F_num;i++) {
      F_clearness_list.add(i, 90);
   }
   //メール頻度初期設定(本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませ
ることになる)
   for (int i=0; i<F_num;i++) {
      mail_frequency_list.add(i, 0);
   }
   //要素数初期設定(本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませること
ことになる)
   for (int i=0; i<F_num;i++) {
      F_ELM_list.add(i, 1);
   }
   //受信メール表示スイッチ初期設定(将来的には前回アプリ終了時にデータを残しておいてそ
れを読み込ませることになると思われる)
   for (int i=0; i<F_num;i++) {
      RM_switch_list.add(i, 0);
   }

   //アイコン情報の作成//()の中身は, アイコンの識別番号, アイコンの色情報, アイコンの名前,
アイコンのアドレス
   Emergent Chara0 = new Emergent(0, color(217, 159, 80), "A", "kei.matsuoka@z2.keio.jp");
   Emergent Chara1 = new Emergent(1, color(80, 159, 217), "B", "mkk0317@gmail.com");
   Emergent Chara2 = new Emergent(2, color(159, 217, 80), "C", "keim0317@hotmail.com");
   Emergent Chara3 = new Emergent(3, color(159, 80, 217), "D", "niconama55t@gmail.com");
   Emergent Chara4 = new Emergent(4, color(217, 80, 159), "E", "yuumeniconico@gmail.com");
   Emergent Chara5 = new Emergent(5, color(80, 217, 159), "F", "keim0317@gmail.com");
```

```
    Chara_list.add(Chara0);
    Chara_list.add(Chara1);
    Chara_list.add(Chara2);
    Chara_list.add(Chara3);
    Chara_list.add(Chara4);
    Chara_list.add(Chara5);
    //形状初期設定(それぞれのキャラクタの創発初期設定を行う)
    for (int i=0 ; i < F_num; i++) {
        Chara_list.get(i).Initialize();
    }
}

void draw() {
    background(255);
    translate(0, 0, 0);

    //カメラ関連の記述
    if (c_theta > 360) c_theta = c_theta - 360;
    if (c_theta < -360 ) c_theta = c_theta + 360;
    positionCamera(c_rho, c_phi, c_theta);          //カメラの位置を変更するためのメソッド, 詳細は
Camera のタブに記述
    camera(camEyeX, camEyeY, camEyeZ, camCentreX, camCentreY, camCentreZ, camUpX,
camUpY, camUpZ);

    //ライト関連の記述
    ambientLight(102, 100, 102);
    directionalLight(255, 255, 255, 1, 1, -1);
    directionalLight(255, 255, 255, -1, 1, 1);

    //5 分間隔でメールボックスの確認を行う
    int check_time_R = (millis()+1000) % 3000000; //受信タイミング用変数
    if (check_time_R >= 0 && check_time_R < 100) {
        checkMail();
    }
    //人との繋がりの深まりの表現・透明度の変化
    int mail_interval = (millis()+1000) % 300000;
    if (mail_interval >= 0 && mail_interval < 10) {
        for (int i=0; i<F_num;i++) {
            if (mail_frequency_list.get(i) > 10) { //一定期間内にメールを相手と 10 通以上送受信してい
たら
                if (F_clearness_list.get(i) <= 240) {
                    F_clearness_list.set(i, F_clearness_list.get(i) + 15 ); //色を濃くする
                }
            }
            else {
                if (F_clearness_list.get(i) >= 15) {
                    F_clearness_list.set(i, F_clearness_list.get(i) - 15 ); //色を薄くする
                }
            }
```

```
            mail_frequency_list.set(i, 0);//0 に戻す
        }
    }
    //要素の消去(思い出の薄れの表現)
    int check_time_D = millis() % 300000;   //半日間隔
    if (check_time_D >= 0 && check_time_D < 50) {
        for (int i=0; i<F_num;i++) {
            if (F_clearness_list.get(i) <= 0) {   //透明になった後で消していく
                Chara_list.get(i).Box_delete();
            }
        }
    }

    //複数のキャラクタが表示される画面の描画
    bg.draw(); //背景の描画
    gie.draw(); //アイコンの描画

    //名前・受信メール内容の描画
    if (View_chara > 0) {
        if (P_NUM_R_list.get(View_chara-1) == 0) {
            pr.view(View_chara-1, View_color);
        }
        if (P_NUM_R_list.get(View_chara-1) > 0) {
            rm.view(View_chara-1, View_color);
        }
    }
}

void stop() {
    super.stop();
}
```

```
////////////////////
/////背景設定/////
////////////////////
public class back_ground {
   ArrayList<piller> pillers = new ArrayList<piller>();   //輪のオブジェクトを格納する配列
   public back_ground() {
   }
   //背景に表示する輪のセットアップ
   //()内の設定は，輪の色，輪の色の透明度，
   //輪の内側の円の半径，輪の外側の円の半径，輪の内側の円の高さ，輪の外側の円の高さ，
回転速度
   //お好みで設定してください
   public void set_up() {
      piller p1 = new piller(color(217, 80, 159), 80, 1300, 1400, 300, 300, -0.01);
      piller p2 = new piller(color(0, 162, 223), 80, 1000, 1100, 350, 350, 0.005);
      piller p3 = new piller(color(159, 217, 80), 80, 400, 500, 500, 500, 0.01);
      piller p4 = new piller(color(217, 159, 80), 80, 600, 700, 240, 240, -0.003);
      piller p5 = new piller(color(80, 217, 159), 80, 500, 600, 540, 540, -0.008);
      piller p6 = new piller(color(217, 100, 80), 230, (F_num-1)*200, (F_num-1)*200+50, 50, 50,
0.007);
      pillers.add(p1);
      pillers.add(p2);
      pillers.add(p3);
      pillers.add(p4);
      pillers.add(p5);
      pillers.add(p6);
   }
   //アイコンが載っているオレンジ色の輪のセットアップ
   public void set_up2() {
      piller p6 = new piller(color(217, 100, 80), 250, (F_num-1)*150, (F_num-1)*150+50, 50, 50,
0.007);
      pillers.set(5, p6);
   }
   public void draw() {
      for (int i=0; i<pillers.size()-1;i++) {
         pillers.get(i).draw();
      }
      pillers.get(5).draw2();
   }
}

public class piller {
   float rot = 0;
   float rot_plus ;
   color c;              //輪の色
   int cl;               //輪の色の透明度
   int up_R ;            //輪の内側の円の半径
   int down_R ;          //輪の外側の円の半径
   int up_Y;             //輪の内側の円の高さ
```

```
    int down_Y;            //輪の外側の円の高さ
    int start = round(random(-20, -12));
    int end = round(random(12, 20));
    public piller(color _c, int _cl, int _up_R, int _down_R, int _up_Y, int _down_Y, float _rot_plus) {
        c = _c;
        cl = _cl;
        up_R = _up_R;
        down_R = _down_R;
        up_Y = _up_Y;
        down_Y = _down_Y;
        rot_plus = _rot_plus;
    }
    //輪の描画メソッド
    public void draw() {
        pushMatrix();
        pushStyle();
        rotateY(rot);
        smooth();
        fill(c, cl);
        noStroke();
        beginShape(TRIANGLE_STRIP);
        for (int i=start;i<end;i++) {
            vertex(up_R*cos(PI*i/40), up_Y, up_R*sin(PI*i/40));
            vertex(down_R*cos(PI*(i-1)/40), down_Y, down_R*sin(PI*(i-1)/40));
        }
        endShape();
        rot += rot_plus;
        popStyle();
        popMatrix();
    }
    //アイコンが載っているオレンジ色の輪のための描画メソッド
     public void draw2() {
        pushMatrix();
        pushStyle();
        rotateY(rot);
        smooth();
        fill(c, cl);
        noStroke();
        beginShape(TRIANGLE_STRIP);
        for (int i=0;i<81;i++) {
            vertex(up_R*cos(PI*i/40), up_Y, up_R*sin(PI*i/40));
            vertex(down_R*cos(PI*(i-1)/40), down_Y, down_R*sin(PI*(i-1)/40));
        }
        endShape();
        rot += rot_plus;
        popStyle();
        popMatrix();
    }
}
```

```
//////////////////////////////////////
/////カメラ位置変更に関する記述/////
//////////////////////////////////////

Float camEyeX = 0.0;                //カメラ自体の X 座標
Float camEyeY = -1000.0;            //カメラ自体の Y 座標
Float camEyeZ = 0.0;                //カメラ自体の Z 座標
Float camCentreX = 0.0;             //カメラの注目位置の X 座標
Float camCentreY = 0.0;             //カメラの注目位置の Y 座標
Float camCentreZ = 0.0;             //カメラの注目位置の Z 座標
Float camUpX = 0.0;                 //画面 X 軸の正負を定義する変数
Float camUpY = 1.0;                 //画面 Y 軸の正負を定義する変数
Float camUpZ = 0.0;                 //画面 Z 軸の正負を定義する変数

//カメラの位置を定義するために用いる変数
float c_phi = 90; // Rotation in YZ plane
float c_theta = -20; // Rotation in XZ plane
float c_rho = 2000; //Distance to camera

//カメラ位置変更メソッド
void positionCamera(float r, float p, float t) {
   //三平方の定理に基づいてカメラ位置を定義
   camEyeX = r * cos(radians(t)) * cos(radians(p));
   camEyeY = r * sin(radians(t));
   camEyeZ = r * cos(radians(t)) * sin(radians(p)) ;

   //カメラ位置が新しく定義されると同時に
   //各アイコンの要素の画面上での位置情報を更新する
   for (int i = 0; i < F_num; i++) {
      Chara_list.get(i).calculateScreenCoordinates();   //各アイコンの要素位置情報メソッドを呼び
出している，詳細は Emergent のタブへ
   }
}
```

```
//////////////////////////////////////////////////
/////アイコンを構成する要素の描画に関する記述/////
//////////////////////////////////////////////////
int tollerance = 50;
int SelectFlag = 0;
int SelectX;
int SelectY;
int SelectZ;
int SelectPID;
int noSelect = 0;
boolean firstSelect = true;

public class E_Box {
  float w, h, d;          //要素の縦, 横, 奥行き
  float px, py, pz;      //要素の 3 次元座標
  color c;                  //要素の色
  int p_num_b;            //引き継がれたアイコン番号
  int col ;
  int count = 0;
  int range = 255;
  int R;

  E_Box(int _p_num_b, float _px, float _py, float _pz) {
    p_num_b = _p_num_b;
    px = _px;
    py = _py;
    pz = _pz;
  }
  public void setSize(float myW, float myH, float myD) {
    w = myW;
    h = myH;
    d = myD;
  }
  void setPosition(float npx, float npy, float npz) {
    px = npx;
    py = npy;
    pz = npz;
  }
  public void setColor(color myC) {
    c = myC;
  }

  public void draw() {
    pushMatrix();
    translate(px, py, pz);
    fill(c, F_clearness_list.get(p_num_b));
    //受信時の色点滅
    if (P_NUM_R_list.get(p_num_b) > 0) {
      if (count >= range) {
```

```
        col = range * 2 - count;
    }
    else {
        col = count;
    }
    if (col == 0 ) {
        fill(red(c)*255, green(c)*255, blue(c)*255, F_clearness_list.get(p_num_b));
    }
    fill(red(c)*255/col, green(c)*255/col, blue(c)*255/col, F_clearness_list.get(p_num_b));
    if (count == range * 2) {
        count = 0;//元に戻す
    }
    count += 15;//カウンターを増やす
}
smooth();
pushStyle();
stroke(c, 90);
if (F_ELM_list.get(p_num_b) >= 0 && F_ELM_list.get(p_num_b) < 20 ) {
    sphereDetail(2);
    sphere(w/2.6);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(2);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <=150 && F_clearness_list.get(p_num_b) > 60) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(3);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) >= 150) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(4);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
    && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
    sphereDetail(2);
    sphere(w/2.6);
```

```
            noStroke();
            sphereDetail(5);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
            && F_clearness_list.get(p_num_b) <= 150 && F_clearness_list.get(p_num_b) > 60) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(6);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
            && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) > 150) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(7);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(8);
            sphere(w/1.5);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 150 && F_clearness_list.get(p_num_b) > 60) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(12);
            sphere(w/1.5);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) > 150) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(20);
            sphere(w/1.5);
        }
        popStyle();
        popMatrix();
    }
}
```

```
//////////////////////////////////////////////////
/////アイコンの形状生成とタッチの判定に関する記述/////
//////////////////////////////////////////////////

//Emergent クラス
public class Emergent {
   //メンバー変数
   float ElmSize =40;              //要素サイズ
   float distance =40;             //立方体における中心間の距離
   int st_r = (F_num-1)*200;
   int icon_height = 150;
   int KSX = 50;                   //形状生成空間 x 軸の要素数
   int KSY = 50;                   //形状生成空間 y 軸の要素数
   int KSZ = 50;                   //形状生成空間 z 軸の要素数
   float halfDis_x = distance*(KSX-1)/2;          //形状生成空間 x 軸における一辺の長さの半分を計算
   float halfDis_y = distance*(KSY-1)/2;          //形状生成空間 y 軸における一辺の長さの半分を計算
   float halfDis_z = distance*(KSZ-1)/2;          //形状生成空間 z 軸における一辺の長さの半分を計算
   int[][][][] MDL = new int[KSX][KSY][KSZ][3];        //形状生成空間を表す配列
   int[][][] MDLnx = new int[KSX][KSY][KSZ]; //次期に生長する要素を格納する配列
   int[] AP = new int[3];                              //頂部座標を格納する配列
   int AP_dx = 0;                  //注目要素が受ける x 軸方向の頂部の力
   int AP_dy = 0;                  //注目要素が受ける y 軸方向の頂部の力
   int AP_dz = 0;                  //注目要素が受ける z 軸方向の頂部の力
   int m;                          //周辺要素番号
   int x, y, z;
   float theta;                    //xy 平面の入力ベクトルの角度 θ
   float phai;                     //z 軸方向の入力ベクトルの角度 φ
   float adtheta;                  //xy 平面の入力ベクトルに加える角度 θ
   float adphai;                   //z 軸方向の入力ベクトルに加える角度 φ
   float d;                        //注目要素-頂部間距離
   float DMAX;                     //頂部最大距離
   float f;                        //1 つの要素が受ける頂部による力
   int zero;                       //入力ベクトルの有無に対するフラグ
   float xmax, ymax, zmax;//3 軸方向における要素と頂部間の最大距離
   float inX, inY, inZ;            //軸方向の入力ベクトル
   float outT;                     //xy 平面の出力ベクトル
   float outP;                     //z 軸方向の出力ベクトル
   int si;                         //x-y 平面における入力ベクトルの離散化番号
   int sj;                         //z 軸方向における入力ベクトルの離散化番号
   int dtrX;                       //x 軸における次期生長要素の発生方向
   int dtrY;                       //y 軸における次期生長要素の発生方向
   int dtrZ;                       //z 軸における次期生長要素の発生方向
   int aa =0;
   int gn_flag = 0;

   int PID = 1;
```

```
    ArrayList<E_Box> boxes = new ArrayList<E_Box>();          //アイコンの要素情報を格納する
配列
    ArrayList<String> mail_info = new ArrayList<String>(); //届いたメールの情報を格納する配列
    float K;                                                  //形状操作パラメータ k
    int[] gn = new int[92];                                  //形状の遺伝子情報を格納する配
列
    int p_num;                                                //アイコンの識別番号
    color box_color;                                         //要素の色情報
    String FName;                                             //アイコンの名前
    String FAddress;                                         //アイコンのアドレス

    //コンストラクタ(初期化)
    Emergent(int _p_num, color _box_color, String _FName, String _FAddress) {
        //メンバ変数の初期設定
        p_num = _p_num;
        box_color = _box_color;
        FName = _FName;
        FAddress = _FAddress;
    }
    //名前設定メソッド(今は使っていない)
    public void SetName(String name) {
        FName = name;
    }
    //アドレス設定メソッド(今は使っていない)
    public void SetAddress(String address) {
        FAddress = address;
    }
    //名前取得メソッド
    public String GetName() {
        return FName;
    }
    //アドレス取得メソッド
    public String GetAddress() {
        return FAddress;
    }
    //着信メール情報の入力メソッド
    //L に送信者の名前, LL に件名, LLL に本文がそれぞれ格納される
    public void in_mail_info(String L, String LL, String LLL) {
        mail_info.add(0, L);
        mail_info.add(1, LL);
        mail_info.add(2, LLL);
    }
    //着信メール情報の出力メソッド
    public String out_mail_info(int i) {
        return mail_info.get(i);
    }

    //初期要素の位置情報の値を返すメソッド
```

```
public float f_elm_x(){
    return (KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num));
}
public float f_elm_y(){
    return (KSY/2)*distance-halfDis_y-icon_height;
}
public float f_elm_z(){
    return (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num));
}

//多様解導出システム一括初期設定メソッド
public void Initialize() {
    Apex_config();      //頂部位置設定
    space_format();     //形状生成空間初期化
    //gn_reader();      //gn 情報の読み込み(ランダムに gn 情報を生成させたい場合はコメントアウトしてください)
    gn_make();          //gn 情報の生成(任意の gn 情報を読み込ませたい場合はコメントアウトしてください)
    ini_elem();         //初期要素配置の設定
    add_ini_elm();      //初期要素配置を MDL 配列に追加
}

//頂部位置設定メソッド
public void Apex_config() {
    AP[0] = int(KSX/2);
    AP[1] = int(KSY/2);
    AP[2] = int(KSZ/2);
    xmax=float(KSX-2)-float(AP[0]);
    ymax=float(KSY-2)-float(AP[1]);
    zmax=float(KSZ-2)-float(AP[2]);
    if (xmax < float(AP[0])) {
        xmax=float(AP[0]);
    }
    if (ymax < float(AP[1])) {
        ymax=float(AP[1]);
    }
    if (zmax < float(AP[2])) {
        zmax=float(AP[2]);
    }
    DMAX=sqrt(xmax*xmax+ymax*ymax+zmax*zmax);    //頂部最大距離の計算
}

//形状生成空間初期化メソッド
public void space_format() {
    for (int i=0; i<KSX; i++) {
        for (int j=0; j<KSY; j++) {
            for (int k=0; k<KSZ; k++) {
                MDL[i][j][k][0] = -5;
            }
```

```
        }
    }
    for (int i=1; i<(KSX-1); i++) {
        for (int j=1; j<(KSY-1); j++) {
            for (int k=1; k<(KSZ-1); k++) {
                MDL[i][j][k][0] = 0;
            }
        }
    }
}

//gn 情報作成メソッド
public void gn_make() {
    for (int i=0; i<92; i++) {
        gn[i] = round(random(8));
    }
}

//初期要素配置メソッド
public void ini_elem() {
    MDL[KSX/2][KSY/2][KSZ/2][0] = 1;
}
//初期要素の boxes への追加
public void add_ini_elm() {
    E_Box box = new E_Box(p_num, (KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
    (KSY/2)*distance-halfDis_y-icon_height,
    (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    box.setSize(ElmSize, ElmSize, ElmSize);
    box.setColor(box_color);
    boxes.add(box);
}


//形状生成メソッド
public void emergent() {
    //形状操作パラメータ変更メソッドの呼び出し
    this.K_change();
    ///// 形状生成アルゴリズム /////
    //形状生成空間内の要素探索//
    for (int x = 1; x < (KSX-1); x ++) {
        for (int y = 1; y < (KSY-1); y ++) {
            for (int z = 1; z < (KSZ-1); z ++) {
                if (MDL[x][y][z][0] >= 1) {

                    //パラメータの初期化//
                    inX=0.0;
                    inY=0.0;
                    inZ=0.0;
                    dtrX=0;
```

```
dtrY=0;
dtrZ=0;
m=0;
gn_flag = 0;

//注目要素周辺の要素の探索//
for (int i=-1; i<2; i++) {
   for (int j=-1; j<2; j++) {
      for (int k=-1; k<2; k++) {
         if (i*9+j*3+k != 0) {
            if (MDL[x+i][y+j][z+k][0] >= 1) {
               //誘導の力計算//
               inX=inX-(K*float(i)*float(gn[m])/sqrt(i*i+j*j+k*k));
               inY=inY-(K*float(j)*float(gn[m])/sqrt(i*i+j*j+k*k));
               inZ=inZ-(K*float(k)*float(gn[m])/sqrt(i*i+j*j+k*k));
            }
            m++;
         }
      }
   }
}

//頂部の力計算//
AP_dx = (AP[0]-x);
AP_dy = (AP[1]-y);
AP_dz = (AP[2]-z);
d=sqrt(AP_dx*AP_dx+AP_dy*AP_dy+AP_dz*AP_dz); //頂部距離算出
if (d != 0.0) {
   f=-(1-K)*(DMAX-d)*12; //頂部支配の力算出
   inX=inX+AP_dx/d*f;      //↓各成分への分解と誘導の力への加算
   inY=inY+AP_dy/d*f;
   inZ=inZ+AP_dz/d*f;
}

//入力ベクトル算出//
if (inX == 0.0 && inY == 0.0) {
   if (inZ == 0.0) {
      zero=1;
      theta=0.0;
      phai=0.0;
   }
   else {
      zero=0;
      theta=0.0;
      if (inZ>0.0) {
         phai=PI/2.0;
      }
      else {
         phai=-(PI/2.0);
```

```
        }
    }
}
else {
    zero=0;
    theta=atan2(inY, inX);
    phai=atan(inZ/sqrt(inX*inX+inY*inY));
}

//入力ベクトルの判定//
for (int i=0; i<=7; i++) {
    if ((float((i-4))*PI/4.0) <= theta && (float((i-3))*PI/4.0) > theta || theta == PI) {
        for (int j=0; j<=3; j++) {
            if ((float((j-2))*PI/4.0) <= phai && (float((j-1))*PI/4.0) > phai) {
                si=i;
                sj=j;
                if (theta == PI) {
                    si=7;
                    sj=j;
                }
            }
        }
        if (phai == PI/2.0) {
            if (theta != PI) {
                si=i;
            }
            sj=3;
        }
    }
}

//出力ベクトルの算出//
if (zero == 1) {
    outT=float(gn[90])*(PI/4.0);
    outP=float(gn[91])*(PI/8.0);
}
else {
    adtheta=float(gn[26+si*4+sj])*(PI/4.0);
    adphai=float(gn[26+si*4+sj+32])*(PI/8.0);
    outT=theta+adtheta;
    outP=phai+adphai;
}
if (outT>PI) {
    outT=outT-(2.0*PI);
}
if (outP>PI/2.0) {
    outP=outP-PI;
}
```

```
//出力ベクトルの判定(曽根式)//
if (6.0*PI/13.0 >= outP && outP > -(6.0*PI/13.0)) {
  if (3.0*PI/8.0 >= outT && outT > -(3.0*PI/8.0)) {
    dtrX=1;
  }
  if ((PI >= outT && outT > 5.0*PI/8.0) || (-(5.0*PI/8.0) >= outT && outT > -PI)) {
    dtrX=-1;
  }
  if (7.0*PI/8.0>=outT && outT>PI/8.0) {
    dtrY=1;
  }
  if (-(PI/8.0) >= outT && outT > -(7.0*PI/8.0)) {
    dtrY=-1;
  }
  if (6.0*PI/13.0 >= outP && outP > 2.0*PI/13.0) {
    dtrZ=1;
  }
  if (-(2.0*PI/13.0) >= outP && outP > -(6.0*PI/13.0)) {
    dtrZ=-1;
  }
}
else if (PI/2.0 >= outP && outP > 6.0*PI/13.0) {
  dtrZ=1;
}
else {
  dtrZ=-1;
}

////////次期発生要素の記録////////
if (MDL[x+dtrX][y+dtrY][z+dtrZ][0] == 0) {
  if (MDLnx[x+dtrX][y+dtrY][z+dtrZ] == 0) {
    MDLnx[x+dtrX][y+dtrY][z+dtrZ] = F_ELM_list.get(p_num) + 1;
    E_Box box = new E_Box(p_num,
(x+dtrX)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
    (y+dtrY)*distance-halfDis_y-icon_height,
    (z+dtrZ)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    box.setSize(ElmSize, ElmSize, ElmSize);
    box.setColor(box_color);
    boxes.add(box);
    F_ELM_list.set(p_num, F_ELM_list.get(p_num) + 1);
    println("boxes.size_" + p_num + "_" + boxes.size() + " F_ELM_" + p_num + "_" +
F_ELM_list.get(p_num));
  }
}
}
}
}
}
//要素の有無を更新//
```

```
    for (int i=1; i<KSX; i++) {
        for (int j=1; j<KSY; j++) {
            for (int k=1; k<KSZ; k++) {
                if (MDLnx[i][j][k] >= 1) {
                    MDL[i][j][k][0] = MDLnx[i][j][k];
                    MDLnx[i][j][k] = 0;
                    gn_flag = 1;
                }
            }
        }
    }
    if (gn_flag == 0) {
        this.gn_make();
        this.emergent();
    }
}

//形状描画メソッド
public void draw() {
    for (int i = 0; i < boxes.size(); i++) {
        boxes.get(i).draw();
    }
}

//形状操作パラメータ変更メソッド
public void K_change() {
    if (F_ELM_list.get(p_num) >= 20 && F_ELM_list.get(p_num) < 150) {
        K = 0.9;
    }
    if (F_ELM_list.get(p_num) < 20) {
        K = 0.1;
    }
    println(K);
}
//要素削除メソッド
public void Box_delete() {
    for (int i=1; i<KSX; i++) {
        for (int j=1; j<KSY; j++) {
            for (int k=1; k<KSZ; k++) {
                if (F_ELM_list.get(p_num) > 1) {                    //要素数が 1 より少なくなら
ないようにする
                    if (MDL[i][j][k][0] == F_ELM_list.get(p_num)) {     //一番新しい要素を見つける
                        MDL[i][j][k][0] = 0;                        //一番新しい要素を消
す
                        boxes.remove(boxes.size()-1);               //一番新しい要素を要
素格納配列からも消す
                        F_ELM_list.set(p_num, F_ELM_list.get(p_num) -1);    //要素数を消した分だけ減
らす
                        k=KSZ;                                      //一個だけ消すた
```

めに for ループを抜ける

```
                    j=KSY;
                    i=KSX;
                    println("boxes.size_" + p_num + "_" + boxes.size() + " F_ELM_" + p_num + "_" +
F_ELM_list.get(p_num));
                }
            }
        }
    }
  }
  }

  //MDL 配列出力用メソッド
  public void Output_MDLdata() {
    PrintWriter output;
    output = createWriter("MDL_data.txt");
    for (int i=0; i<KSX; i++) {
      for (int j=0; j<KSY; j++) {
        for (int k=0; k<KSZ; k++) {
          output.flush();
          output.print(MDL[i][j][k][0]+" ");
        }
      }
    }
    output.close();
  }

  public void set_up() {
    st_r = (F_num-1)*200;
  }

  //位置修正メソッド(新しいアイコンが追加されたときに使う)
  public void translate_box() {
    for (int i=1; i<KSX; i++) {
      for (int j=1; j<KSY; j++) {
        for (int k=1; k<KSZ; k++) {
          if (MDL[i][j][k][0] > 0) {

boxes.get(MDL[i][j][k][0]-1).setPosition((i)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
              (j)*distance-halfDis_y-icon_height,
              (k)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
          }
        }
      }
    }
    println(ElmSize);
  }

  void calculateScreenCoordinates() {
```

```
    //Caclulates the corresponding x and y screen coordinates for all lattice points and stores then in the lattice array

    /* for (int i = 0; i < KSX-1; i++) {
      for (int j = 0; j < KSY-1; j++) {
      for (int k = 0; k < KSZ-1; k++) {
      //Store the current screen coordinates of each point in the lattice array
      MDL[i][j][k][1] = (int)screenX(i*distance-halfDis_x, j*distance-halfDis_y, k*distance-halfDis_z);
      MDL[i][j][k][2] = (int)screenY(i*distance-halfDis_x, j*distance-halfDis_y, k*distance-halfDis_z);
      }
      }
      }*/
    MDL[KSX/2][KSY/2][KSZ/2][1] = (int)screenX((KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)), (KSY/2)*distance-halfDis_y-icon_height, (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    MDL[KSX/2][KSY/2][KSZ/2][2] = (int)screenY((KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)), (KSY/2)*distance-halfDis_y-icon_height, (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
  }



  void preDrawSelect(int x, int y) {
    //Default to nothing being selected
    int preDrawSelection = -1;

    //Iterate through array and return array
    for (int i = 0; i < KSX-1 && SelectFlag==0; i++) {
      for (int j = 0; j < KSY-1 && SelectFlag==0; j++) {
        for (int k = 0; k < KSZ-1 && SelectFlag==0; k++) {
          if (MDL[i][j][k][0] >= 1) {
            //If the mouse is within the tollerance range of a lattice point
            if (x >= MDL[i][j][k][1] - tollerance && x <= MDL[i][j][k][1] + tollerance && y >= MDL[i][j][k][2] - tollerance && y <= MDL[i][j][k][2] + tollerance && SelectFlag==0) {

              if (firstSelect == true) {
                println("MDL"+i+" "+j+" "+k+"_"+MDL[i][j][k][0]);
                SelectX = i;
                SelectY = j;
                SelectZ = k;
                SelectPID = MDL[i][j][k][0];
                println("SelectPID_"+ SelectPID);
                println("PID_" + PID);
                MDL[i][j][k][0] = PID+1;
                println("MDL"+i+" "+j+" "+k+"_"+MDL[i][j][k][0]);
```

```
                    firstSelect = false;

                    View_chara = p_num + 1;
                    View_color = box_color;

                    P_NUM_S = p_num;
                    println(P_NUM_S);
                }
                SelectFlag = 1;
            }
            else {
                if (firstSelect == false) {
                    MDL[SelectX][SelectY][SelectZ][0] = SelectPID;
                    println("PIDreturn");
                    firstSelect = true;
                    View_chara = 0;
                }
            }
        }
    }
  }
}

//gn 情報読み込み用メソッド
public void gn_reader() {
    try {
        FileReader in = new FileReader("C:/Users/Mr.k/Documents/Processing/Kizuna_mail/gndata/gn_data_" + p_num + ".txt");
        BufferedReader br = new BufferedReader(in);
        String line;
        while ( (line = br.readLine ()) != null) {
            //System.out.print(line);
            String lines[] = split(line, "  ");
            int counter = 0;
            for (int i=0; i<92; i++) {
                gn[i] = int(lines[counter]);
                counter++;
            }
        }
        br.close();
        in.close();
    }
    catch(FileNotFoundException e) {
        System.out.println(e);
    }
    catch(IOException e) {
        System.out.println(e);
```

```
        }
        for (int i=0; i<92; i++) {
            print(gn[i]);
        }
        println(" ");
    }
}
```

```
//////////////////////////////////////////////////
/////画面上に表示するアイコン，文字等の記述/////
//////////////////////////////////////////////////

public class GraphicInterface_E {
   int state = 0;
   int e_try = 0;
   int starter = 0;
   public GraphicInterface_E() {
   }
   public void draw() {
      //形状の描画
      if (starter == 0) {
         while (e_try < 0) {
            for (int j=0 ; j < F_num ; j++) {
               Chara_list.get(j).emergent();
            }
            e_try ++;
         }
         starter = 1;
      }
      if (New_switch == 1) {
         e_try = 0;
         while (e_try < 0) {
            Chara_list.get(F_num-1).emergent();
            e_try ++;
         }
         New_switch = 0;
      }
      //送受信による形状生成
      for (int i=0 ; i < F_num ; i++) {
         if (F_ELM_list.get(i) >= 1 && F_ELM_list.get(i) < 150) {
            if (ER_switch == 1) {
               for (int j=0; j<F_num; j++) {
                  if (P_NUM_R_list.get(j) > 0) {
                     Chara_list.get(j).emergent();
                  }
               }
               ER_switch = 0;
            }
            if (ES_switch == 1) {
               Chara_list.get(P_NUM_S).emergent();
               Chara_list.get(P_NUM_S).emergent();
               ES_switch = 0;
            }
         }
      }
      for (int i=0 ; i < F_num ; i++) {
         Chara_list.get(i).draw();
```

```
        }
    }
}

public class RM_view {
    PGraphics mg;
    public RM_view() {
        mg=createGraphics(width, height, P3D);
    }
    //受信メール表示メソッド
    public void view(int i, color c) {
        PFont font = createFont("MS Gothic", 24); //フォントを変換
        textMode(SCREEN);                                  //テキストの描画方法の設定
        textAlign(LEFT);                                   //テキストの位置合わせ設定
        textFont(font);                                    //フォントを設定
        fill(100);                                         //文字色
        text(Chara_list.get(i).out_mail_info(0), 210, 100, width-300, height); //From 表示
        text(Chara_list.get(i).out_mail_info(1), 210, 150, width-300, height); //Subject 表示
        text(Chara_list.get(i).out_mail_info(2), 210, 200, width-300, height); //message 表示
        fill(c);
        text("[FROM]", 100, 100, width-300, height); //From 表示
        text("[SUBJECT]", 100, 150, width-300, height); //Subject 表示
        text("[MESSAGE]", 100, 200, width-300, height); //message 表示
    }
}

public class profile_view {
    public profile_view() {
    }
    //プロフィール表示メソッド
    public void view(int i, color c) {
        pushMatrix();
        PFont font = createFont("MS Gothic", 24); //フォントを変換
        textMode(SCREEN);                                  //テキストの描画方法の設定
        textAlign(CENTER);                                 //テキストの位置合わせ設定
        textFont(font);                                    //フォントを設定
        fill(c);                                           //文字色
        text(Chara_list.get(i).GetName(),
        screenX(Chara_list.get(i).f_elm_x(), Chara_list.get(i).f_elm_y(), Chara_list.get(i).f_elm_z()),
        screenY(Chara_list.get(i).f_elm_x(),  Chara_list.get(i).f_elm_y(),  Chara_list.get(i).f_elm_z())  -
100); //From 表示
        popMatrix();
    }
}
```

```
//////////////////////////////////
/////メールの認証に関する記述/////
//////////////////////////////////

import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;

public class Auth extends Authenticator {
   public Auth() {
      super();
   }
   public PasswordAuthentication getPasswordAuthentication() {
      String username, password;
      username = "testkv001@gmail.com";//user ID
      password = "kei870317";//password
      System.out.println("authenticating. . ");
      return new PasswordAuthentication(username, password);
   }
}
```

```
//////////////////////////////////////
/////メール編集画面に関する記述/////
//////////////////////////////////////

import java.awt.*;
import javax.swing.*;

void Mailgui() {

////////////////////////////////////////////////////////////////////////////////
////////////////////////
   JPanel panel = new JPanel();

   BoxLayout layout = new BoxLayout(panel, BoxLayout.PAGE_AXIS); //PAGE_AXIS＝行が進む
方向, Y_AXIS 与同一
   panel.setLayout(layout);

   JButton to_button = new JButton("TO");
   JButton cc_button = new JButton("CC");
   JButton bcc_button = new JButton("BCC");
   JButton subj_button = new JButton("件名");
   JButton m_button = new JButton("本文");

   JTextField text1 = new JTextField(Chara_list.get(P_NUM_S).GetAddress());
   JTextField text2 = new JTextField();
   JTextField text3 = new JTextField();
   JTextField text4 = new JTextField();

   //テキスト枠のサイズ設定
   text1.setPreferredSize(new Dimension(768, 20));
   text2.setPreferredSize(new Dimension(768, 20));
   text3.setPreferredSize(new Dimension(768, 20));
   text4.setPreferredSize(new Dimension(768, 20));

   panel.add(to_button);
   panel.add(Box.createGlue());
   panel.add(text1);
   panel.add(cc_button);
   panel.add(Box.createGlue());
   panel.add(text2);
   panel.add(bcc_button);
   panel.add(Box.createGlue());
   panel.add(text3);
   panel.add(subj_button);
   panel.add(Box.createGlue());
   panel.add(text4);

   panel.add(m_button);
   panel.add(Box.createGlue());
```

```
    JTextArea text5 = new JTextArea();
    text5.setLineWrap(true);              //折り返しコード
    text5.setWrapStyleWord(true);         //折り返しコード
    text5.setPreferredSize(new Dimension(768, 200));
    panel.add(text5);

    String selectvalues[] = {
        "送信"
    };

    int r = JOptionPane.showOptionDialog(null,
    panel, //メッセージ
    "送信画面", //ダイアログのタイトル
    JOptionPane.YES_NO_OPTION, //オプションタイプ
    JOptionPane.PLAIN_MESSAGE, //メッセージタイプ
    null, //アイコンなし
    selectvalues, //ボタンの配列
    selectvalues[0]);

    to = Chara_list.get(P_NUM_S).GetAddress();          //TO
    cc = text2.getText();                                 //CC
    bcc = text3.getText();                                //BCC
    subj = text4.getText();                              //件名
    maintxt = text5.getText();                           //本文

    if(r == JOptionPane.CLOSED_OPTION){
        ES_switch = 0;
    }
    else{
    ES_switch = 1;
    }

////////////////////////////////////////////////////////////////////////////////
////////////////////////
}
```

```
//////////////////////////////
/////メール受信に関する記述/////
//////////////////////////////

PMailReceiver receiver;
PMailSender sender;
void checkMail() {

  receiver=new PMailReceiver(this, "pop.gmail.com", " testkv001@gmail.com ", "kei870317");
  receiver.update();

  println(receiver.getMessageCount() + " total messages.");
  PMessage[] messages=receiver.getUnreadMessages();
  for (int i=0;i<messages.length;i++) {
    //受信相手の判別
    String list[] = split(messages[i].getFrom(), ' ');
    println(list[1]);
    for (int j=0; j<F_num; j++) {
      String F_list = "<" + Chara_list.get(j).GetAddress() + ">";
      println(F_list);
      if (list[1].equals(F_list)) { //注意：String の比較の際は==ではなく.eqals()を使う。
        println("アドレス合致");
        ER_switch = 1;
        P_NUM_R_list.set(j, P_NUM_R_list.get(j) +1 );
        println(P_NUM_R_list.get(j));
      }
      //メール情報の格納
      Chara_list.get(j).in_mail_info(Chara_list.get(j).GetName(),          messages[i].getSubject(),
messages[i].getMessage());
    }

    println("----------------------");
    println("Message # " + (i+1));
    println("From: " + messages[i].getFrom());
    println("Subject: " + messages[i].getSubject());
    println("Message:");
    println(messages[i].getMessage());
  }
  receiver.quit();
}
```

```
//////////////////////////////
/////メール送信に関する記述/////
//////////////////////////////

String to;
String cc;
String bcc;
String subj;
String maintxt;

void sendMail() {
   // Create a session
   String host="smtp.gmail.com";
   Properties props=new Properties();

   // SMTP Session
   props.put("mail.transport.protocol", "smtp");
   props.put("mail.smtp.host", host);
   props.put("mail.smtp.port", "587");
   props.put("mail.smtp.auth", "true");
   // We need TTLS, which gmail requires
   props.put("mail.smtp.starttls.enable", "true");

   // Create a session
   Session session = Session.getDefaultInstance(props, new Auth());

   try{
      // Make a new message
      MimeMessage message = new MimeMessage(session);

      // Who is this message from
      message.setFrom(new InternetAddress("testkv001@gmail.com", "kei870317"));

      Mailgui();

      if(ES_switch == 1){
      // Who is this message to (we could do fancier things like make a list or add CC's)
      message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to, false));
      message.setRecipients(Message.RecipientType.CC, InternetAddress.parse(cc, false));
      message.setRecipients(Message.RecipientType.BCC, InternetAddress.parse(bcc, false));

      // Subject and body
      message.setSubject(subj);
      message.setText(maintxt);

      // We can do more here, set the date, the headers, etc.
      //error here
      Transport.send(message);
      println("Mail sent!");
```

```
        }
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}
```

```
//////////////////////////////////
/////マウス操作とキーボード操作/////
//////////////////////////////////

void mouseClicked() {
  if (mouseButton == LEFT&&SelectFlag==1) {
    noSelect = 1;
    sendMail();
    P_NUM_R_list.set(P_NUM_S, 0);
  }
  println(SelectFlag);
  noSelect = 0;
}

void mouseDragged(){
  //右ドラッグしながらY軸方向に動かすと拡大，縮小する
  if (mouseButton == RIGHT){
    c_rho -= pmouseY - mouseY;
  }

  //左ドラッグしながらX軸方向に動かすと，カメラが回転する
  if (mouseButton == LEFT){
    if ( mouseY > screenY(0, 100, 0)) {
      c_phi -= pmouseX - mouseX;
    }
    else {
      c_phi += pmouseX - mouseX;
    }
  }
}

void mouseMoved(){
  //カーソルが動くと，アイコンの選択判定が行われる
  SelectFlag=0;
  if (noSelect == 0) {
    for (int i = 0; i < F_num; i++ ) {
      Chara_list.get(i).preDrawSelect(mouseX, mouseY);
    }
  }
}

void keyPressed() {
  //「f」キーを押すと，新規アイコン追加メソッドが呼び出される
  if (key=='f') {
    add_chara();
  }
  //「r」キーを押すと，メールの受信チェックを任意で行うことが出来る
  if (key=='r') {
    checkMail();
```

```
}

/////以下，デモ用のキーボード操作/////
if (key=='t') {
   P_NUM_R_list.set(1, 1);
}
if (key=='q') {
   saveFrame("line-####.png");
}
if (key=='0') {
   if (F_ELM_list.get(0) >= 1 && F_ELM_list.get(0) < 150) {
      Chara_list.get(0).emergent();
   }
}
if (key=='1') {
   if (F_ELM_list.get(1) >= 1 && F_ELM_list.get(1) < 150) {
      Chara_list.get(1).emergent();
   }
}
if (key=='2') {
   if (F_ELM_list.get(2) >= 1 && F_ELM_list.get(2) < 150) {
      Chara_list.get(2).emergent();
   }
}
if (key=='3') {
   if (F_ELM_list.get(3) >= 1 && F_ELM_list.get(3) < 150) {
      Chara_list.get(3).emergent();
   }
}
if (key=='4') {
   if (F_ELM_list.get(4) >= 1 && F_ELM_list.get(4) < 150) {
      Chara_list.get(4).emergent();
   }
}
if (key=='5') {
   if (F_ELM_list.get(5) >= 1 && F_ELM_list.get(5) < 150) {
      Chara_list.get(5).emergent();
   }
}
if (key=='u') {
   if (F_clearness_list.get(P_NUM_S) <= 240) {
      F_clearness_list.set(P_NUM_S, F_clearness_list.get(P_NUM_S) + 15 );
   }
   println(F_clearness_list.get(P_NUM_S));
}
if (key=='d') {
   if (F_clearness_list.get(P_NUM_S) >= 15) {
      F_clearness_list.set(P_NUM_S, F_clearness_list.get(P_NUM_S) - 15 );
   }
```

```
        println(F_clearness_list.get(P_NUM_S));
    }

    if (key=='e') {
        Chara_list.get(P_NUM_S).Box_delete();
    }
}

//新規アイコン追加メソッド(txt データ入力バージョン. 将来的には専用の UI を作って直接入力できるようにしたほうがいい)
public void add_chara() {
    String currentPath = null;                    //現在選択中のファイルパスの変数
    String loadPath = selectInput();              //ファイル選択画面を表示し選択したファイルパス取得
    if (loadPath == null) {                       //ファイルパスが空の場合
        println("No file was selected...");   //「ファイルが選ばれてない」メッセージを出力
        loadPath=currentPath;                         //ファイルパスを前回のファイルパスにする
    }
    else {                                        //ファイルパスが選択された場合
        String ext = loadPath.substring(loadPath.indexOf('.') + 1);     //ファイルパスのドット以降の文字列を取得(拡張子名を取得)
        if (ext.equals("txt")) {                                        //拡張子が「txt」なら
            String lines[] = loadStrings(loadPath);                     //選択ファイルパスのテキストを取り込み
            currentPath=loadPath;                                       //現在選択中のファイルパスを更新
            println(currentPath);                                       //現在選択中のファイルパスを出力
            F_num = F_num + 1;
            println("num-ok");
            //既存キャラの移動
            for (int i=0; i<F_num-1; i++) {
                Chara_list.get(i).set_up();
                Chara_list.get(i).translate_box();
            }
            println("t-ok");

            //lines に格納された String データをもとに新しいキャラを作成
            Emergent New_Chara = new Emergent(F_num-1, color(round(random(80, 150)), round(random(80, 150)), round(random(80, 150))), lines[0], lines[1]);
            println("e-ok");
            Chara_list.add(New_Chara);
            println("add-ok");
            P_NUM_R_list.add(0);
            F_clearness_list.add(60);
            mail_frequency_list.add(0);
            F_ELM_list.add(1);
            RM_switch_list.add(0);
            Chara_list.get(F_num-1).Initialize();
            New_switch = 1;
```

```
            bg.set_up2();
        }
        //else {                                    //拡張子が「txt」ではないとき
            //println("Not txt file.");            //「txt ファイルではない」と出力
        //}
    }
}
```

A.2 2 次プロトタイプ ソースコード

```
////////////////////
/////メイン関数/////
////////////////////

import com.sun.opengl.util.*;
import javax.media.opengl.*;
import processing.opengl.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;
import java.io.File.*;

int ScrSizeX = 1024;                    //ウィンドウサイズ(x 軸)
int ScrSizeY = 768;                     //ウィンドウサイズ(y 軸)
int  F_num  =  6;                          //アイコンの数(最初に複数個表示させたければ複数
個を代入)
int sequencer = 0;                      //画面の切り替え変数
int P_NUM_S = 0 ;                        //送信した相手を確認するための変数
int ER_switch = 0;                      //受信確認時, 受信メールがあった場合、形状生成に移
るための変数
int ES_switch = 0;                      //送信確認時, 形状生成に移るための変数
int New_switch = 0;                     //
int View_chara = 0;                     //
//int header = 0;                          //ヘッダー数取得用変数
color View_color ;                      //プロフィールや
ArrayList<Emergent> Chara_list = new ArrayList<Emergent>(); //F_num の数だけのアイコンのオ
ブジェクトを格納する配列
ArrayList<Integer> F_ELM_list = new ArrayList<Integer>(); //各キャラクタの要素数を格納する配
列
ArrayList<Integer> F_clearness_list = new ArrayList<Integer>(); //各キャラクタの透明度を入れる
配列
ArrayList<Integer> mail_frequency_list = new ArrayList<Integer>(); //一定期間内のメール送受信
回数を入れておく配列
ArrayList<Integer> P_NUM_R_list = new ArrayList<Integer>(); //受信確認時に登録された相手か
ら何通メールが来ていたか入れる配列
ArrayList<Integer> RM_switch_list = new ArrayList<Integer>(); //どのキャラクタが受信しているか
を判定する数値を格納する配列

int[] data_year_list = new int [F_num];//どのキャラクタがどれだけ送受信しているかを年単位で計
測する数値を格納する配列
int[] data_month_list = new int [F_num];//どのキャラクタがどれだけ送受信しているかを月単位で
計測する数値を格納する配列
int[] data_week_list = new int [F_num];//どのキャラクタがどれだけ送受信しているかを週単位で計
測する数値を格納する配列
int weekday=1;
int monthday=1;
int yearday=1;
```

```
int yearcount=0;
int monthcount=0;
int weekcount=0;

//オブジェクトの名称設定
GraphicInterface_E gie;
back_ground bg;
RM_view rm;
profile_view pr;


void setup() {
   size(ScrSizeX, ScrSizeY, OPENGL); //スクリーンサイズの定義
   colorMode(RGB, 255);                    //カラーモードの定義
   hint(ENABLE_OPENGL_4X_SMOOTH);      //4 倍速アンチエイリアスの定義
   frameRate(60);                    //描画フレームレート

   //オブジェクトのインスタンス化
   gie = new GraphicInterface_E();
   bg = new back_ground();
   rm = new RM_view();
   pr = new profile_view();

   //背景のセットアップ
   bg.set_up();                              //内容は Background のタブ参照

   //受信メール数の初期設定（0 通にしておく）
   for (int i=0; i<F_num;i++) {
      P_NUM_R_list.add(i, 0);
   }
   //透明度初期設定（本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませること
ことになる）
   for (int i=0; i < F_num;i++) {
      F_clearness_list.add(i, 90);
   }
   //メール頻度初期設定（本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませ
ることになる）
   for (int i=0; i<F_num;i++) {
      mail_frequency_list.add(i, 0);
   }
   //要素数初期設定（本来なら前回アプリ終了時にデータを残しておいてそれを読み込ませること
ことになる）
   for (int i=0; i<F_num;i++) {
      F_ELM_list.add(i, 1);
   }
   //受信メール表示スイッチ初期設定（将来的には前回アプリ終了時にデータを残しておいてそ
それを読み込ませることになると思われる）
   for (int i=0; i<F_num;i++) {
      RM_switch_list.add(i, 0);
```

```
  }/*
  //メール送受信数ヘッダーカウント用蓄積データ配列
   for (int i=0; i<F_num; i++) {
   data_year_list.add(i,dy);
   data_month_list.add(i,dm);
   data_week_list.add(i,dw);
   }*/

  //アイコン情報の作成//()の中身は，アイコンの識別番号，アイコンの色情報，アイコンの名前，
アイコンのアドレス
  Emergent Chara0 = new Emergent(0, color(217, 159, 80), "A", "kei.matsuoka@z2.keio.jp");
  Emergent Chara1 = new Emergent(1, color(80, 159, 217), "B", "mkk0317@gmail.com");
  Emergent Chara2 = new Emergent(2, color(159, 217, 80), "C", "keim0317@hotmail.com");
  Emergent Chara3 = new Emergent(3, color(159, 80, 217), "D", "niconama55t@gmail.com");
  Emergent Chara4 = new Emergent(4, color(217, 80, 159), "E", "yuumeniconico@gmail.com");
  Emergent Chara5 = new Emergent(5, color(80, 217, 159), "F", "keim0317@gmail.com");

  Chara_list.add(Chara0);
  Chara_list.add(Chara1);
  Chara_list.add(Chara2);
  Chara_list.add(Chara3);
  Chara_list.add(Chara4);
  Chara_list.add(Chara5);
  //形状初期設定(それぞれのキャラクタの創発初期設定を行う)
  for (int i=0 ; i < F_num; i++) {
     Chara_list.get(i).Initialize();
  }
}

void draw() {
  background(255);
  translate(0, 0, 0);

  //カメラ関連の記述
  if (c_theta > 360) c_theta = c_theta - 360;
  if (c_theta < -360 ) c_theta = c_theta + 360;
  positionCamera(c_rho, c_phi, c_theta);          //カメラの位置を変更するためのメソッド，詳細は
Camera のタブに記述
  camera(camEyeX, camEyeY, camEyeZ, camCentreX, camCentreY, camCentreZ, camUpX,
camUpY, camUpZ);

  //ライト関連の記述
  ambientLight(102, 100, 102);
  directionalLight(255, 255, 255, 1, 1, -1);
  directionalLight(255, 255, 255, -1, 1, 1);

  //5 分間隔でメールボックスの確認を行う
  int check_time_R = (millis()+1000) % 3000000; //受信タイミング用変数
  if (check_time_R >= 0 && check_time_R < 100) {
```

```
        checkMail();
    }
    //人との繋がりの深まりの表現・透明度の変化
    int mail_interval = (millis()+1000) % 300000;
    if (mail_interval >= 0 && mail_interval < 10) {
        for (int i=0; i<F_num;i++) {
            if (mail_frequency_list.get(i) > 10) { //一定期間内にメールを相手と 10 通以上送受信してい
たら
                if (F_clearness_list.get(i) <= 240) {
                    F_clearness_list.set(i, F_clearness_list.get(i) + 15 ); //色を濃くする
                }
            }
            else {
                if (F_clearness_list.get(i) >= 15) {
                    F_clearness_list.set(i, F_clearness_list.get(i) - 15 ); //色を薄くする
                }
            }
            mail_frequency_list.set(i, 0);//0 に戻す
        }
    }
    //要素の消去(思い出の薄れの表現)
    int check_time_D = millis() % 300000;   //半日間隔
    if (check_time_D >= 0 && check_time_D < 50) {
        for (int i=0; i<F_num;i++) {
            if (F_clearness_list.get(i) <= 0) {   //透明になった後で消していく
                Chara_list.get(i).Box_delete();
            }
        }
    }

    //複数のキャラクタが表示される画面の描画
    bg.draw(); //背景の描画
    gie.draw(); //アイコンの描画

    //名前・受信メール内容の描画
    if (View_chara > 0) {
        if (P_NUM_R_list.get(View_chara-1) == 0) {
            pr.view(View_chara-1, View_color);
        }
        if (P_NUM_R_list.get(View_chara-1) > 0) {
            rm.view(View_chara-1, View_color);
        }
    }
}

void stop() {
    super.stop();
}
public class back_ground {
```

```
   ArrayList<piller> pillers = new ArrayList<piller>();   //輪のオブジェクトを格納する配列
////////////////////
/////背景設定/////
////////////////////

   public back_ground() {
   }
   //背景に表示する輪のセットアップ
   //()内の設定は，輪の色，輪の色の透明度，
   //輪の内側の円の半径，輪の外側の円の半径，輪の内側の円の高さ，輪の外側の円の高さ，
回転速度
   //お好みで設定してください
   public void set_up() {
      piller p1 = new piller(color(217, 80, 159), 80, 1300, 1400, 300, 300, -0.01);
      piller p2 = new piller(color(0, 162, 223), 80, 1000, 1100, 350, 350, 0.005);
      piller p3 = new piller(color(159, 217, 80), 80, 400, 500, 500, 500, 0.01);
      piller p4 = new piller(color(217, 159, 80), 80, 600, 700, 240, 240, -0.003);
      piller p5 = new piller(color(80, 217, 159), 80, 500, 600, 540, 540, -0.008);
      piller p6 = new piller(color(217, 100, 80), 230, (F_num-1)*200, (F_num-1)*200+50, 50, 50,
0.007);
      pillers.add(p1);
      pillers.add(p2);
      pillers.add(p3);
      pillers.add(p4);
      pillers.add(p5);
      pillers.add(p6);
   }
   //アイコンが載っているオレンジ色の輪のセットアップ
   public void set_up2() {
      piller p6 = new piller(color(217, 100, 80), 250, (F_num-1)*150, (F_num-1)*150+50, 50, 50,
0.007);
      pillers.set(5, p6);
   }
   public void draw() {
      for (int i=0; i<pillers.size()-1;i++) {
         pillers.get(i).draw();
      }
      pillers.get(5).draw2();
   }
}

public class piller {
   float rot = 0;
   float rot_plus ;
   color c;              //輪の色
   int cl;               //輪の色の透明度
   int up_R ;            //輪の内側の円の半径
   int down_R ;          //輪の外側の円の半径
   int up_Y;             //輪の内側の円の高さ
```

```
  int down_Y;          //輪の外側の円の高さ
  int start = round(random(-20, -12));
  int end = round(random(12, 20));
  public piller(color _c, int _cl, int _up_R, int _down_R, int _up_Y, int _down_Y, float _rot_plus) {
    c = _c;
    cl = _cl;
    up_R = _up_R;
    down_R = _down_R;
    up_Y = _up_Y;
    down_Y = _down_Y;
    rot_plus = _rot_plus;
  }
  //輪の描画メソッド
  public void draw() {
    pushMatrix();
    pushStyle();
    rotateY(rot);
    smooth();
    fill(c, cl);
    noStroke();
    beginShape(TRIANGLE_STRIP);
    for (int i=start;i<end;i++) {
      vertex(up_R*cos(PI*i/40), up_Y, up_R*sin(PI*i/40));
      vertex(down_R*cos(PI*(i-1)/40), down_Y, down_R*sin(PI*(i-1)/40));
    }
    endShape();
    rot += rot_plus;
    popStyle();
    popMatrix();
  }
  //アイコンが載っているオレンジ色の輪のための描画メソッド
   public void draw2() {
    pushMatrix();
    pushStyle();
    rotateY(rot);
    smooth();
    fill(c, cl);
    noStroke();
    beginShape(TRIANGLE_STRIP);
    for (int i=0;i<81;i++) {
      vertex(up_R*cos(PI*i/40), up_Y, up_R*sin(PI*i/40));
      vertex(down_R*cos(PI*(i-1)/40), down_Y, down_R*sin(PI*(i-1)/40));
    }
    endShape();
    rot += rot_plus;
    popStyle();
    popMatrix();
  }
}
```

```
//////////////////////////////////////
/////カメラ位置変更に関する記述/////
//////////////////////////////////////

Float camEyeX = 0.0;                //カメラ自体の X 座標
Float camEyeY = -1000.0;            //カメラ自体の Y 座標
Float camEyeZ = 0.0;                //カメラ自体の Z 座標
Float camCentreX = 0.0;             //カメラの注目位置の X 座標
Float camCentreY = 0.0;             //カメラの注目位置の Y 座標
Float camCentreZ = 0.0;             //カメラの注目位置の Z 座標
Float camUpX = 0.0;                 //画面 X 軸の正負を定義する変数
Float camUpY = 1.0;                 //画面 Y 軸の正負を定義する変数
Float camUpZ = 0.0;                 //画面 Z 軸の正負を定義する変数

//カメラの位置を定義するために用いる変数
float c_phi = 90; // Rotation in YZ plane
float c_theta = -20; // Rotation in XZ plane
float c_rho = 2000; //Distance to camera

//カメラ位置変更メソッド
void positionCamera(float r, float p, float t) {
   //三平方の定理に基づいてカメラ位置を定義
   camEyeX = r * cos(radians(t)) * cos(radians(p));
   camEyeY = r * sin(radians(t));
   camEyeZ = r * cos(radians(t)) * sin(radians(p)) ;

   //カメラ位置が新しく定義されると同時に
   //各アイコンの要素の画面上での位置情報を更新する
   for (int i = 0; i < F_num; i++) {
      Chara_list.get(i).calculateScreenCoordinates();   //各アイコンの要素位置情報メソッドを呼び
出している，詳細は Emergent のタブへ
   }
}
```

```
//////////////////////////////////////////////////
/////アイコンを構成する要素の描画に関する記述/////
//////////////////////////////////////////////////
int tollerance = 50;
int SelectFlag = 0;
int SelectX;
int SelectY;
int SelectZ;
int SelectPID;
int noSelect = 0;
boolean firstSelect = true;

public class E_Box {
  float w, h, d;          //要素の縦, 横, 奥行き
  float px, py, pz;      //要素の 3 次元座標
  color c;                  //要素の色
  int p_num_b;            //引き継がれたアイコン番号
  int col ;
  int count = 0;
  int range = 255;
  int R;

  E_Box(int _p_num_b, float _px, float _py, float _pz) {
    p_num_b = _p_num_b;
    px = _px;
    py = _py;
    pz = _pz;
  }
  public void setSize(float myW, float myH, float myD) {
    w = myW;
    h = myH;
    d = myD;
  }
  void setPosition(float npx, float npy, float npz) {
    px = npx;
    py = npy;
    pz = npz;
  }
  public void setColor(color myC) {
    c = myC;
  }

  public void draw() {
    pushMatrix();
    translate(px, py, pz);
    fill(c, F_clearness_list.get(p_num_b));
    //受信時の色点滅
    if (P_NUM_R_list.get(p_num_b) > 0) {
      if (count >= range) {
```

```
        col = range * 2 - count;
    }
    else {
        col = count;
    }
    if (col == 0 ) {
        fill(red(c)*255, green(c)*255, blue(c)*255, F_clearness_list.get(p_num_b));
    }
    fill(red(c)*255/col, green(c)*255/col, blue(c)*255/col, F_clearness_list.get(p_num_b));
    if (count == range * 2) {
        count = 0;//元に戻す
    }
    count += 15;//カウンターを増やす
}
smooth();
pushStyle();
stroke(c, 90);
if (F_ELM_list.get(p_num_b) >= 0 && F_ELM_list.get(p_num_b) < 20 ) {
    sphereDetail(2);
    sphere(w/2.6);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(2);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <=150 && F_clearness_list.get(p_num_b) > 60) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(3);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 20 && F_ELM_list.get(p_num_b) < 40
    && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) >= 150) {
    sphereDetail(2);
    sphere(w/2.6);
    noStroke();
    sphereDetail(4);
    sphere(w/2);
}
if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
    && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
    sphereDetail(2);
    sphere(w/2.6);
```

```
            noStroke();
            sphereDetail(5);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
            && F_clearness_list.get(p_num_b) <= 150 && F_clearness_list.get(p_num_b) > 60) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(6);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 40 && F_ELM_list.get(p_num_b) < 100
            && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) > 150) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(7);
            sphere(w/1.7);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 60 && F_clearness_list.get(p_num_b) >= 0) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(8);
            sphere(w/1.5);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 150 && F_clearness_list.get(p_num_b) > 60) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(12);
            sphere(w/1.5);
        }
        if (F_ELM_list.get(p_num_b) >= 100
            && F_clearness_list.get(p_num_b) <= 255 && F_clearness_list.get(p_num_b) > 150) {
            sphereDetail(2);
            sphere(w/2.6);
            noStroke();
            sphereDetail(20);
            sphere(w/1.5);
        }
        popStyle();
        popMatrix();
    }
}
```

```
//////////////////////////////////////////////////
/////アイコンの形状生成とタッチの判定に関する記述/////
//////////////////////////////////////////////////

//Emergent クラス
public class Emergent {
   //メンバー変数
   float ElmSize =40;              //要素サイズ
   float distance =40;             //立方体における中心間の距離
   int st_r = (F_num-1)*200;
   int icon_height = 150;
   int KSX = 50;                   //形状生成空間 x 軸の要素数
   int KSY = 50;                   //形状生成空間 y 軸の要素数
   int KSZ = 50;                   //形状生成空間 z 軸の要素数
   float halfDis_x = distance*(KSX-1)/2;          //形状生成空間 x 軸における一辺の長さの半分を計算
   float halfDis_y = distance*(KSY-1)/2;          //形状生成空間 y 軸における一辺の長さの半分を計算
   float halfDis_z = distance*(KSZ-1)/2;          //形状生成空間 z 軸における一辺の長さの半分を計算
   int[][][][] MDL = new int[KSX][KSY][KSZ][3];        //形状生成空間を表す配列
   int[][][] MDLnx = new int[KSX][KSY][KSZ]; //次期に生長する要素を格納する配列
   int[] AP = new int[3];                              //頂部座標を格納する配列
   int AP_dx = 0;                  //注目要素が受ける x 軸方向の頂部の力
   int AP_dy = 0;                  //注目要素が受ける y 軸方向の頂部の力
   int AP_dz = 0;                  //注目要素が受ける z 軸方向の頂部の力
   int m;                          //周辺要素番号
   int x, y, z;
   float theta;                    //xy 平面の入力ベクトルの角度 θ
   float phai;                     //z 軸方向の入力ベクトルの角度 φ
   float adtheta;                  //xy 平面の入力ベクトルに加える角度 θ
   float adphai;                   //z 軸方向の入力ベクトルに加える角度 φ
   float d;                        //注目要素-頂部間距離
   float DMAX;                     //頂部最大距離
   float f;                        //1 つの要素が受ける頂部による力
   int zero;                       //入力ベクトルの有無に対するフラグ
   float xmax, ymax, zmax;//3 軸方向における要素と頂部間の最大距離
   float inX, inY, inZ;            //軸方向の入力ベクトル
   float outT;                     //xy 平面の出力ベクトル
   float outP;                     //z 軸方向の出力ベクトル
   int si;                         //x-y 平面における入力ベクトルの離散化番号
   int sj;                         //z 軸方向における入力ベクトルの離散化番号
   int dtrX;                       //x 軸における次期生長要素の発生方向
   int dtrY;                       //y 軸における次期生長要素の発生方向
   int dtrZ;                       //z 軸における次期生長要素の発生方向
   int aa =0;
   int gn_flag = 0;

   int PID = 1;
```

```
    ArrayList<E_Box> boxes = new ArrayList<E_Box>();          //アイコンの要素情報を格納する
配列
    ArrayList<String> mail_info = new ArrayList<String>(); //届いたメールの情報を格納する配列
    float K;                                                  //形状操作パラメータ k
    int[] gn = new int[92];                                  //形状の遺伝子情報を格納する配
列
    int p_num;                                                //アイコンの識別番号
    color box_color;                                         //要素の色情報
    String FName;                                             //アイコンの名前
    String FAddress;                                         //アイコンのアドレス

    //コンストラクタ(初期化)
    Emergent(int _p_num, color _box_color, String _FName, String _FAddress) {
        //メンバ変数の初期設定
        p_num = _p_num;
        box_color = _box_color;
        FName = _FName;
        FAddress = _FAddress;
    }
    //名前設定メソッド(今は使っていない)
    public void SetName(String name) {
        FName = name;
    }
    //アドレス設定メソッド(今は使っていない)
    public void SetAddress(String address) {
        FAddress = address;
    }
    //名前取得メソッド
    public String GetName() {
        return FName;
    }
    //アドレス取得メソッド
    public String GetAddress() {
        return FAddress;
    }
    //着信メール情報の入力メソッド
    //L に送信者の名前, LL に件名, LLL に本文がそれぞれ格納される
    public void in_mail_info(String L, String LL, String LLL) {
        mail_info.add(0, L);
        mail_info.add(1, LL);
        mail_info.add(2, LLL);
    }
    //着信メール情報の出力メソッド
    public String out_mail_info(int i) {
        return mail_info.get(i);
    }

    //初期要素の位置情報の値を返すメソッド
```

```
public float f_elm_x(){
    return (KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num));
}
public float f_elm_y(){
    return (KSY/2)*distance-halfDis_y-icon_height;
}
public float f_elm_z(){
    return (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num));
}

//多様解導出システム一括初期設定メソッド
public void Initialize() {
    Apex_config();      //頂部位置設定
    space_format();     //形状生成空間初期化
    //gn_reader();      //gn 情報の読み込み(ランダムに gn 情報を生成させたい場合はコメントアウトしてください)
    gn_make();          //gn 情報の生成(任意の gn 情報を読み込ませたい場合はコメントアウトしてください)
    ini_elem();         //初期要素配置の設定
    add_ini_elm();      //初期要素配置を MDL 配列に追加
}

//頂部位置設定メソッド
public void Apex_config() {
    AP[0] = int(KSX/2);
    AP[1] = int(KSY/2);
    AP[2] = int(KSZ/2);
    xmax=float(KSX-2)-float(AP[0]);
    ymax=float(KSY-2)-float(AP[1]);
    zmax=float(KSZ-2)-float(AP[2]);
    if (xmax < float(AP[0])) {
        xmax=float(AP[0]);
    }
    if (ymax < float(AP[1])) {
        ymax=float(AP[1]);
    }
    if (zmax < float(AP[2])) {
        zmax=float(AP[2]);
    }
    DMAX=sqrt(xmax*xmax+ymax*ymax+zmax*zmax);     //頂部最大距離の計算
}

//形状生成空間初期化メソッド
public void space_format() {
    for (int i=0; i<KSX; i++) {
        for (int j=0; j<KSY; j++) {
            for (int k=0; k<KSZ; k++) {
                MDL[i][j][k][0] = -5;
            }
```

```
        }
    }
    for (int i=1; i<(KSX-1); i++) {
        for (int j=1; j<(KSY-1); j++) {
            for (int k=1; k<(KSZ-1); k++) {
                MDL[i][j][k][0] = 0;
            }
        }
    }
}

//gn 情報作成メソッド
public void gn_make() {
    for (int i=0; i<92; i++) {
        gn[i] = round(random(8));
    }
}

//初期要素配置メソッド
public void ini_elem() {
    MDL[KSX/2][KSY/2][KSZ/2][0] = 1;
}
//初期要素の boxes への追加
public void add_ini_elm() {
    E_Box box = new E_Box(p_num, (KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
    (KSY/2)*distance-halfDis_y-icon_height,
    (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    box.setSize(ElmSize, ElmSize, ElmSize);
    box.setColor(box_color);
    boxes.add(box);
}


//形状生成メソッド
public void emergent() {
    //形状操作パラメータ変更メソッドの呼び出し
    this.K_change();
    ///// 形状生成アルゴリズム /////
    //形状生成空間内の要素探索//
    for (int x = 1; x < (KSX-1); x ++) {
        for (int y = 1; y < (KSY-1); y ++) {
            for (int z = 1; z < (KSZ-1); z ++) {
                if (MDL[x][y][z][0] >= 1) {

                    //パラメータの初期化//
                    inX=0.0;
                    inY=0.0;
                    inZ=0.0;
                    dtrX=0;
```

```
dtrY=0;
dtrZ=0;
m=0;
gn_flag = 0;

//注目要素周辺の要素の探索//
for (int i=-1; i<2; i++) {
   for (int j=-1; j<2; j++) {
      for (int k=-1; k<2; k++) {
         if (i*9+j*3+k != 0) {
            if (MDL[x+i][y+j][z+k][0] >= 1) {
               //誘導の力計算//
               inX=inX-(K*float(i)*float(gn[m])/sqrt(i*i+j*j+k*k));
               inY=inY-(K*float(j)*float(gn[m])/sqrt(i*i+j*j+k*k));
               inZ=inZ-(K*float(k)*float(gn[m])/sqrt(i*i+j*j+k*k));
            }
            m++;
         }
      }
   }
}

//頂部の力計算//
AP_dx = (AP[0]-x);
AP_dy = (AP[1]-y);
AP_dz = (AP[2]-z);
d=sqrt(AP_dx*AP_dx+AP_dy*AP_dy+AP_dz*AP_dz); //頂部距離算出
if (d != 0.0) {
   f=-(1-K)*(DMAX-d)*12; //頂部支配の力算出
   inX=inX+AP_dx/d*f;      //↓各成分への分解と誘導の力への加算
   inY=inY+AP_dy/d*f;
   inZ=inZ+AP_dz/d*f;
}

//入力ベクトル算出//
if (inX == 0.0 && inY == 0.0) {
   if (inZ == 0.0) {
      zero=1;
      theta=0.0;
      phai=0.0;
   }
   else {
      zero=0;
      theta=0.0;
      if (inZ>0.0) {
         phai=PI/2.0;
      }
      else {
         phai=-(PI/2.0);
```

```
        }
    }
}
else {
    zero=0;
    theta=atan2(inY, inX);
    phai=atan(inZ/sqrt(inX*inX+inY*inY));
}

//入力ベクトルの判定//
for (int i=0; i<=7; i++) {
    if ((float((i-4))*PI/4.0) <= theta && (float((i-3))*PI/4.0) > theta || theta == PI) {
        for (int j=0; j<=3; j++) {
            if ((float((j-2))*PI/4.0) <= phai && (float((j-1))*PI/4.0) > phai) {
                si=i;
                sj=j;
                if (theta == PI) {
                    si=7;
                    sj=j;
                }
            }
        }
        if (phai == PI/2.0) {
            if (theta != PI) {
                si=i;
            }
            sj=3;
        }
    }
}

//出力ベクトルの算出//
if (zero == 1) {
    outT=float(gn[90])*(PI/4.0);
    outP=float(gn[91])*(PI/8.0);
}
else {
    adtheta=float(gn[26+si*4+sj])*(PI/4.0);
    adphai=float(gn[26+si*4+sj+32])*(PI/8.0);
    outT=theta+adtheta;
    outP=phai+adphai;
}
if (outT>PI) {
    outT=outT-(2.0*PI);
}
if (outP>PI/2.0) {
    outP=outP-PI;
}
```

```
//出力ベクトルの判定(曽根式)//
if (6.0*PI/13.0 >= outP && outP > -(6.0*PI/13.0)) {
  if (3.0*PI/8.0 >= outT && outT > -(3.0*PI/8.0)) {
    dtrX=1;
  }
  if ((PI >= outT && outT > 5.0*PI/8.0) || (-(5.0*PI/8.0) >= outT && outT > -PI)) {
    dtrX=-1;
  }
  if (7.0*PI/8.0>=outT && outT>PI/8.0) {
    dtrY=1;
  }
  if (-(PI/8.0) >= outT && outT > -(7.0*PI/8.0)) {
    dtrY=-1;
  }
  if (6.0*PI/13.0 >= outP && outP > 2.0*PI/13.0) {
    dtrZ=1;
  }
  if (-(2.0*PI/13.0) >= outP && outP > -(6.0*PI/13.0)) {
    dtrZ=-1;
  }
}
else if (PI/2.0 >= outP && outP > 6.0*PI/13.0) {
  dtrZ=1;
}
else {
  dtrZ=-1;
}

////////次期発生要素の記録////////
if (MDL[x+dtrX][y+dtrY][z+dtrZ][0] == 0) {
  if (MDLnx[x+dtrX][y+dtrY][z+dtrZ] == 0) {
    MDLnx[x+dtrX][y+dtrY][z+dtrZ] = F_ELM_list.get(p_num) + 1;
    E_Box box = new E_Box(p_num,
(x+dtrX)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
    (y+dtrY)*distance-halfDis_y-icon_height,
    (z+dtrZ)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    box.setSize(ElmSize, ElmSize, ElmSize);
    box.setColor(box_color);
    boxes.add(box);
    F_ELM_list.set(p_num, F_ELM_list.get(p_num) + 1);
    println("boxes.size_" + p_num + "_" + boxes.size() + " F_ELM_" + p_num + "_" +
F_ELM_list.get(p_num));
  }
}
}
}
}
}
//要素の有無を更新//
```

```
        for (int i=1; i<KSX; i++) {
            for (int j=1; j<KSY; j++) {
                for (int k=1; k<KSZ; k++) {
                    if (MDLnx[i][j][k] >= 1) {
                        MDL[i][j][k][0] = MDLnx[i][j][k];
                        MDLnx[i][j][k] = 0;
                        gn_flag = 1;
                    }
                }
            }
        }
        if (gn_flag == 0) {
            this.gn_make();
            this.emergent();
        }
    }

    //形状描画メソッド
    public void draw() {
        for (int i = 0; i < boxes.size(); i++) {
            boxes.get(i).draw();
        }
    }

    //形状操作パラメータ変更メソッド
    public void K_change() {
        if (F_ELM_list.get(p_num) >= 20 && F_ELM_list.get(p_num) < 150) {
            K = 0.9;
        }
        if (F_ELM_list.get(p_num) < 20) {
            K = 0.1;
        }
        println(K);
    }
    //要素削除メソッド
    public void Box_delete() {
        for (int i=1; i<KSX; i++) {
            for (int j=1; j<KSY; j++) {
                for (int k=1; k<KSZ; k++) {
                    if (F_ELM_list.get(p_num) > 1) {                    //要素数が 1 より少なくなら
ないようにする
                        if (MDL[i][j][k][0] == F_ELM_list.get(p_num)) {     //一番新しい要素を見つける
                            MDL[i][j][k][0] = 0;                        //一番新しい要素を消
す
                            boxes.remove(boxes.size()-1);               //一番新しい要素を要
素格納配列からも消す
                            F_ELM_list.set(p_num, F_ELM_list.get(p_num) -1);    //要素数を消した分だけ減
らす
                            k=KSZ;                                      //一個だけ消すた
```

めに for ループを抜ける

```
                    j=KSY;
                    i=KSX;
                    println("boxes.size_" + p_num + "_" + boxes.size() + " F_ELM_" + p_num + "_" +
F_ELM_list.get(p_num));
                }
            }
        }
    }
  }
  }

  //MDL 配列出力用メソッド
  public void Output_MDLdata() {
    PrintWriter output;
    output = createWriter("MDL_data.txt");
    for (int i=0; i<KSX; i++) {
      for (int j=0; j<KSY; j++) {
        for (int k=0; k<KSZ; k++) {
          output.flush();
          output.print(MDL[i][j][k][0]+" ");
        }
      }
    }
    output.close();
  }

  public void set_up() {
    st_r = (F_num-1)*200;
  }

  //位置修正メソッド(新しいアイコンが追加されたときに使う)
  public void translate_box() {
    for (int i=1; i<KSX; i++) {
      for (int j=1; j<KSY; j++) {
        for (int k=1; k<KSZ; k++) {
          if (MDL[i][j][k][0] > 0) {

boxes.get(MDL[i][j][k][0]-1).setPosition((i)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)),
            (j)*distance-halfDis_y-icon_height,
            (k)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
          }
        }
      }
    }
    println(ElmSize);
  }

  void calculateScreenCoordinates() {
```

```
    //Caclulates the corresponding x and y screen coordinates for all lattice points and stores then in the lattice array

    /* for (int i = 0; i < KSX-1; i++) {
      for (int j = 0; j < KSY-1; j++) {
      for (int k = 0; k < KSZ-1; k++) {
      //Store the current screen coordinates of each point in the lattice array
      MDL[i][j][k][1] = (int)screenX(i*distance-halfDis_x, j*distance-halfDis_y, k*distance-halfDis_z);
      MDL[i][j][k][2] = (int)screenY(i*distance-halfDis_x, j*distance-halfDis_y, k*distance-halfDis_z);
      }
      }
      }*/
    MDL[KSX/2][KSY/2][KSZ/2][1] = (int)screenX((KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)), (KSY/2)*distance-halfDis_y-icon_height, (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
    MDL[KSX/2][KSY/2][KSZ/2][2] = (int)screenY((KSX/2)*distance-halfDis_x+st_r*(cos((PI*2/F_num)*p_num)), (KSY/2)*distance-halfDis_y-icon_height, (KSZ/2)*distance-halfDis_z+st_r*(sin((PI*2/F_num)*p_num)));
  }



  void preDrawSelect(int x, int y) {
    //Default to nothing being selected
    int preDrawSelection = -1;

    //Iterate through array and return array
    for (int i = 0; i < KSX-1 && SelectFlag==0; i++) {
      for (int j = 0; j < KSY-1 && SelectFlag==0; j++) {
        for (int k = 0; k < KSZ-1 && SelectFlag==0; k++) {
          if (MDL[i][j][k][0] >= 1) {
            //If the mouse is within the tollerance range of a lattice point
            if (x >= MDL[i][j][k][1] - tollerance && x <= MDL[i][j][k][1] + tollerance && y >= MDL[i][j][k][2] - tollerance && y <= MDL[i][j][k][2] + tollerance && SelectFlag==0) {

              if (firstSelect == true) {
                println("MDL"+i+" "+j+" "+k+"_"+MDL[i][j][k][0]);
                SelectX = i;
                SelectY = j;
                SelectZ = k;
                SelectPID = MDL[i][j][k][0];
                println("SelectPID_"+ SelectPID);
                println("PID_" + PID);
                MDL[i][j][k][0] = PID+1;
                println("MDL"+i+" "+j+" "+k+"_"+MDL[i][j][k][0]);
```

```
                    firstSelect = false;

                    View_chara = p_num + 1;
                    View_color = box_color;

                    P_NUM_S = p_num;
                    println(P_NUM_S);
                }
                SelectFlag = 1;
            }
            else {
                if (firstSelect == false) {
                    MDL[SelectX][SelectY][SelectZ][0] = SelectPID;
                    println("PIDreturn");
                    firstSelect = true;
                    View_chara = 0;
                }
            }
        }
    }
  }
}

//gn 情報読み込み用メソッド
public void gn_reader() {
    try {
        FileReader in = new FileReader("C:/Users/Mr.k/Documents/Processing/Kizuna_mail/gndata/gn_data_" + p_num + ".txt");
        BufferedReader br = new BufferedReader(in);
        String line;
        while ( (line = br.readLine ()) != null) {
            //System.out.print(line);
            String lines[] = split(line, "  ");
            int counter = 0;
            for (int i=0; i<92; i++) {
                gn[i] = int(lines[counter]);
                counter++;
            }
        }
        br.close();
        in.close();
    }
    catch(FileNotFoundException e) {
        System.out.println(e);
    }
    catch(IOException e) {
        System.out.println(e);
```

```
        }
        for (int i=0; i<92; i++) {
            print(gn[i]);
        }
        println(" ");
    }
}
```

```
//////////////////////////////////////////////
/////画面上に表示するアイコン，文字等の記述/////
//////////////////////////////////////////////

public class GraphicInterface_E {
  int state = 0;
  int e_try = 0;
  int starter = 0;
  public GraphicInterface_E() {
  }
  public void draw() {
    //形状の描画
    if (starter == 0) {
      while (e_try < 0) {
        for (int j=0 ; j < F_num ; j++) {
          Chara_list.get(j).emergent();
        }
        e_try ++;
      }
      starter = 1;
    }
    if (New_switch == 1) {
      e_try = 0;
      while (e_try < 0) {
        Chara_list.get(F_num-1).emergent();
        e_try ++;
      }
      New_switch = 0;
    }
    //送受信による形状生成
    for (int i=0 ; i < F_num ; i++) {
      if (F_ELM_list.get(i) >= 1 && F_ELM_list.get(i) < 150) {
        if (ER_switch == 1) {
          for (int j=0; j<F_num; j++) {
            if (P_NUM_R_list.get(j) > 0) {
              Chara_list.get(j).emergent();
            }
          }
          ER_switch = 0;
        }
        if (ES_switch == 1) {
          Chara_list.get(P_NUM_S).emergent();
          Chara_list.get(P_NUM_S).emergent();
          ES_switch = 0;
        }
      }
    }
    for (int i=0 ; i < F_num ; i++) {
      Chara_list.get(i).draw();
```

```
        }
    }
}

public class RM_view {
    PGraphics mg;
    public RM_view() {
        mg=createGraphics(width, height, P3D);
    }
    //受信メール表示メソッド
    public void view(int i, color c) {
        PFont font = createFont("MS Gothic", 24); //フォントを変換
        textMode(SCREEN);                                  //テキストの描画方法の設定
        textAlign(LEFT);                                   //テキストの位置合わせ設定
        textFont(font);                                    //フォントを設定
        fill(100);                                         //文字色
        text(Chara_list.get(i).out_mail_info(0), 210, 100, width-300, height); //From 表示
        text(Chara_list.get(i).out_mail_info(1), 210, 150, width-300, height); //Subject 表示
        text(Chara_list.get(i).out_mail_info(2), 210, 200, width-300, height); //message 表示
        fill(c);
        text("[FROM]", 100, 100, width-300, height); //From 表示
        text("[SUBJECT]", 100, 150, width-300, height); //Subject 表示
        text("[MESSAGE]", 100, 200, width-300, height); //message 表示
    }
}

public class profile_view {
    public profile_view() {
    }
    //プロフィール表示メソッド
    public void view(int i, color c) {
        int yhdr = data_year_list[i];
        int mhdr = data_month_list[i];
        int whdr = data_week_list[i];

        if (hour()==0 && minute() == 0 && second() ==0) {
            weekday++;
            monthday++;
            yearday++;
        }
        if (weekday==7) {
            weekcount++;
            weekday=1;
        }
        if (monthday==30) {
            monthcount++;
            monthday=1;
        }
        if (yearday==365) {
```

```
        yearcount++;
        yearday=1;
    }
    pushMatrix();
    PFont font = createFont("MS Gothic", 24); //フォントを変換
    textMode(SCREEN);                                  //テキストの描画方法の設定
    textAlign(LEFT);                                   //テキストの位置合わせ設定
    textFont(font);                                    //フォントを設定
    fill(c);                                           //文字色
    text(Chara_list.get(i).GetName() + "の連絡累積回数", width*3/4, 50, width-300, height); //
累積連絡回数
    if (yearcount==0) {
        text("A Year : "+ yhdr + " 回", width*3/4, 100, width-300, height); //
        text("A Month: " + mhdr + " 回", width*3/4, 150, width-300, height); //
        text("A Week : " + whdr + " 回", width*3/4, 200, width-300, height); //
        if (monthcount==0) {
            text("A Year : "+ yhdr + " 回", width*3/4, 100, width-300, height); //
            text("A Month: " + mhdr + " 回", width*3/4, 150, width-300, height); //
            text("A Week : " + whdr + " 回", width*3/4, 200, width-300, height); //
            if (weekcount==0) {
                text("A Year : "+ yhdr + " 回", width*3/4, 100, width-300, height); //
                text("A Month: " + mhdr + " 回", width*3/4, 150, width-300, height); //
                text("A Week : " + whdr + " 回", width*3/4, 200, width-300, height); //
            }
            else if (weekcount==1) {
                data_week_list[i]=0;
                weekcount=0;
            }
        }
        else if (monthcount==1) {
            data_month_list[i] = 0;
            monthcount=0;
        }
    }
    else if (yearcount==1) {
        data_year_list[i] = 0;
        yearcount=0;
    }
    text("A Year : "+ yhdr + " 回", width*3/4, 100, width-300, height); //
    text("A Month: " + mhdr + " 回", width*3/4, 150, width-300, height); //
    text("A Week : " + whdr + " 回", width*3/4, 200, width-300, height); //
    /*
    else if (daycount>=366) {
        text("A Year : "+ data_year_list[i], width*3/4, 100, width-300, height); //
        text("A Month: " + data_month_list[i], width*3/4, 150, width-300, height); //
        text("A Week : " + data_week_list[i], width*3/4, 200, width-300, height); //
        daycount=0;
    }
```

```
      text(Chara_list.get(i).GetName(),
        screenX(Chara_list.get(i).f_elm_x(), Chara_list.get(i).f_elm_y(), Chara_list.get(i).f_elm_z()),
        screenY(Chara_list.get(i).f_elm_x(),  Chara_list.get(i).f_elm_y(),  Chara_list.get(i).f_elm_z())  -
100); //From 表示
        */
      popMatrix();
   }
}
```

```
//////////////////////////////////
/////メールの認証に関する記述/////
//////////////////////////////////

import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;

public class Auth extends Authenticator {
   public Auth() {
      super();
   }
   public PasswordAuthentication getPasswordAuthentication() {
      String username, password;
      username = "testkv001@gmail.com";//user ID
      password = "kei870317";//password
      System.out.println("authenticating. . ");
      return new PasswordAuthentication(username, password);
   }
}
```

```
//////////////////////////////////////
/////メール編集画面に関する記述/////
//////////////////////////////////////

import java.awt.*;
import javax.swing.*;

void Mailgui() {

////////////////////////////////////////////////////////////////////////////////
////////////////////////
   JPanel panel = new JPanel();

   BoxLayout layout = new BoxLayout(panel, BoxLayout.PAGE_AXIS); //PAGE_AXIS＝行が進む
方向, Y_AXIS 与同一
   panel.setLayout(layout);

   JButton to_button = new JButton("TO");
   JButton cc_button = new JButton("CC");
   JButton bcc_button = new JButton("BCC");
   JButton subj_button = new JButton("件名");
   JButton m_button = new JButton("本文");

   JTextField text1 = new JTextField(Chara_list.get(P_NUM_S).GetAddress());
   JTextField text2 = new JTextField();
   JTextField text3 = new JTextField();
   JTextField text4 = new JTextField();

   //テキスト枠のサイズ設定
   text1.setPreferredSize(new Dimension(768, 20));
   text2.setPreferredSize(new Dimension(768, 20));
   text3.setPreferredSize(new Dimension(768, 20));
   text4.setPreferredSize(new Dimension(768, 20));

   panel.add(to_button);
   panel.add(Box.createGlue());
   panel.add(text1);
   panel.add(cc_button);
   panel.add(Box.createGlue());
   panel.add(text2);
   panel.add(bcc_button);
   panel.add(Box.createGlue());
   panel.add(text3);
   panel.add(subj_button);
   panel.add(Box.createGlue());
   panel.add(text4);

   panel.add(m_button);
   panel.add(Box.createGlue());
```

```
    JTextArea text5 = new JTextArea();
    text5.setLineWrap(true);              //折り返しコード
    text5.setWrapStyleWord(true);         //折り返しコード
    text5.setPreferredSize(new Dimension(768, 200));
    panel.add(text5);

    String selectvalues[] = {
        "送信"
    };

    int r = JOptionPane.showOptionDialog(null,
    panel, //メッセージ
    "送信画面", //ダイアログのタイトル
    JOptionPane.YES_NO_OPTION, //オプションタイプ
    JOptionPane.PLAIN_MESSAGE, //メッセージタイプ
    null, //アイコンなし
    selectvalues, //ボタンの配列
    selectvalues[0]);

    to = Chara_list.get(P_NUM_S).GetAddress();          //TO
    cc = text2.getText();                                 //CC
    bcc = text3.getText();                                //BCC
    subj = text4.getText();                              //件名
    maintxt = text5.getText();                           //本文

    if(r == JOptionPane.CLOSED_OPTION){
        ES_switch = 0;
    }
    else{
    ES_switch = 1;
    }

////////////////////////////////////////////////////////////////////////////////
////////////////////////
}
```

```
//////////////////////////////
/////メール受信に関する記述/////
//////////////////////////////

PMailReceiver receiver;
PMailSender sender;
void checkMail() {

  receiver=new PMailReceiver(this, "imap.gmail.com", "testkv001@gmail.com", "kei870317");
  receiver.update();

  println(receiver.getMessageCount() + " total messages.");
  PMessage[] messages=receiver.getUnreadMessages();
  for (int i=0;i<messages.length;i++) {
    //受信相手の判別
    String list[] = split(messages[i].getFrom(), ' ');
    println(list[1]);
    for (int j=0; j<F_num; j++) {
      String F_list = "<" + Chara_list.get(j).GetAddress() + ">";
      println(F_list);
      if (list[1].equals(F_list)) { //注意:String の比較の際は==ではなく.eqals()を使う。
        println("アドレス合致");
        ER_switch = 1;
        P_NUM_R_list.set(j, P_NUM_R_list.get(j) +1 );
        println(P_NUM_R_list.get(j));
        //            header++;
        data_year_list[j]=data_year_list[j]+1;
        data_month_list[j]=data_month_list[j]+1;
        data_week_list[j]=data_week_list[j]+1;
        /*
        data_year_list.add(j, dy++);
         data_month_list.add(j, dm++);
         data_week_list.add(j, dw++);
         */
      }
      //メール情報の格納
      Chara_list.get(j).in_mail_info(Chara_list.get(j).GetName(),          messages[i].getSubject(),
messages[i].getMessage());
    }

    println("----------------------");
    println("Message # " + (i+1));
    println("From: " + messages[i].getFrom());
    println("Subject: " + messages[i].getSubject());
    println("Message:");
    println(messages[i].getMessage());
  }
  receiver.quit();
}
```

```
//////////////////////////////
/////メール送信に関する記述/////
//////////////////////////////

String to;
String cc;
String bcc;
String subj;
String maintxt;

void sendMail() {
   // Create a session
   String host="smtp.gmail.com";
   Properties props=new Properties();

   // SMTP Session
   props.put("mail.transport.protocol", "smtp");
   props.put("mail.smtp.host", host);
   props.put("mail.smtp.port", "587");
   props.put("mail.smtp.auth", "true");
   // We need TTLS, which gmail requires
   props.put("mail.smtp.starttls.enable", "true");

   // Create a session
   Session session = Session.getDefaultInstance(props, new Auth());

   try {
      // Make a new message
      MimeMessage message = new MimeMessage(session);

      // Who is this message from
      message.setFrom(new InternetAddress("testkv001@gmail.com", "kei870317"));

      Mailgui();

      if (ES_switch == 1) {
         // Who is this message to (we could do fancier things like make a list or add CC's)
         message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to, false));
         message.setRecipients(Message.RecipientType.CC, InternetAddress.parse(cc, false));
         message.setRecipients(Message.RecipientType.BCC, InternetAddress.parse(bcc, false));

         // Subject and body
         message.setSubject(subj);
         message.setText(maintxt);

         // We can do more here, set the date, the headers, etc.
         //error here
         Transport.send(message);
         println("Mail sent!");
```

```
        //      header++;
        data_year_list[F_num]=data_year_list[F_num]+1;
        data_month_list[F_num]=data_month_list[F_num]+1;
        data_week_list[F_num]=data_week_list[F_num]+1;
        /*          data_year_list.add(F_num, dy++);
          data_month_list.add(F_num, dm++);
          data_week_list.add(F_num, dw++);*/
      }
    }
    catch(Exception e) {
      e.printStackTrace();
    }
}
```

```
//////////////////////////////////
/////マウス操作とキーボード操作////
//////////////////////////////////

void mouseClicked() {
  if (mouseButton == LEFT&&SelectFlag==1) {
    noSelect = 1;
    sendMail();
    P_NUM_R_list.set(P_NUM_S, 0);
  }
  println(SelectFlag);
  noSelect = 0;
}

void mouseDragged() {
  //右ドラッグしながらY軸方向に動かすと拡大，縮小する
  if (mouseButton == RIGHT) {
    c_rho -= pmouseY - mouseY;
  }

  //左ドラッグしながらX軸方向に動かすと，カメラが回転する
  if (mouseButton == LEFT) {
    if ( mouseY > screenY(0, 100, 0)) {
      c_phi -= pmouseX - mouseX;
    }
    else {
      c_phi += pmouseX - mouseX;
    }
  }
}

void mouseMoved() {
  //カーソルが動くと，アイコンの選択判定が行われる
  SelectFlag=0;
  if (noSelect == 0) {
    for (int i = 0; i < F_num; i++ ) {
      Chara_list.get(i).preDrawSelect(mouseX, mouseY);
    }
  }
}

void keyPressed() {
  //「f」キーを押すと，新規アイコン追加メソッドが呼び出される
  if (key=='f') {
    add_chara();
  }
  //「r」キーを押すと，メールの受信チェックを任意で行うことが出来る
  if (key=='r') {
    checkMail();
```

```
}

/////以下，デモ用のキーボード操作/////
if (key=='t') {
   P_NUM_R_list.set(1, 1);
}
if (key=='q') {
   //saveFrame("line-####.png");
   weekday++;
   monthday++;
   yearday++;
}
if (key=='w') {
   saveFrame("line-####.png");
}
if (key=='0') {
   if (F_ELM_list.get(0) >= 1 && F_ELM_list.get(0) < 150) {
      Chara_list.get(0).emergent();
      //         header++;
      data_year_list[0]=data_year_list[0]+1;
      data_month_list[0]=data_month_list[0]+1;
      data_week_list[0]=data_week_list[0]+1;
   }
}
if (key=='1') {
   if (F_ELM_list.get(1) >= 1 && F_ELM_list.get(1) < 150) {
      Chara_list.get(1).emergent();
      //         header++;
      data_year_list[1]=data_year_list[1]+1;
      data_month_list[1]=data_month_list[1]+1;
      data_week_list[1]=data_week_list[1]+1;
   }
}
if (key=='2') {
   if (F_ELM_list.get(2) >= 1 && F_ELM_list.get(2) < 150) {
      Chara_list.get(2).emergent();
      //         header++;
      data_year_list[2]=data_year_list[2]+1;
      data_month_list[2]=data_month_list[2]+1;
      data_week_list[2]=data_week_list[2]+1;
   }
}
if (key=='3') {
   if (F_ELM_list.get(3) >= 1 && F_ELM_list.get(3) < 150) {
      Chara_list.get(3).emergent();
      //         header++;
      data_year_list[3]=data_year_list[3]+1;
      data_month_list[3]=data_month_list[3]+1;
      data_week_list[3]=data_week_list[3]+1;
```

```
        }
    }
    if (key=='4') {
        if (F_ELM_list.get(4) >= 1 && F_ELM_list.get(4) < 150) {
            Chara_list.get(4).emergent();
            //          header++;
            data_year_list[4]=data_year_list[4]+1;
            data_month_list[4]=data_month_list[4]+1;
            data_week_list[4]=data_week_list[4]+1;
        }
    }
    if (key=='5') {
        if (F_ELM_list.get(5) >= 1 && F_ELM_list.get(5) < 150) {
            Chara_list.get(5).emergent();
            //          header++;
            data_year_list[5]=data_year_list[5]+1;
            data_month_list[5]=data_month_list[5]+1;
            data_week_list[5]=data_week_list[5]+1;
        }
    }
    if (key=='u') {
        if (F_clearness_list.get(P_NUM_S) <= 240) {
            F_clearness_list.set(P_NUM_S, F_clearness_list.get(P_NUM_S) + 15 );
        }
        println(F_clearness_list.get(P_NUM_S));
    }
    if (key=='d') {
        if (F_clearness_list.get(P_NUM_S) >= 15) {
            F_clearness_list.set(P_NUM_S, F_clearness_list.get(P_NUM_S) - 15 );
        }
        println(F_clearness_list.get(P_NUM_S));
    }

    if (key=='e') {
        Chara_list.get(P_NUM_S).Box_delete();
    }
}

//新規アイコン追加メソッド(txt データ入力バージョン. 将来的には専用の UI を作って直接入力できるようにしたほうがいい)
public void add_chara() {
    String currentPath = null;                  //現在選択中のファイルパスの変数
    String loadPath = selectInput();            //ファイル選択画面を表示し選択したファイルパス取得
    if (loadPath == null) {                     //ファイルパスが空の場合
        println("No file was selected...");    //「ファイルが選ばれてない」メッセージを出力
        loadPath=currentPath;                   //ファイルパスを前回のファイルパスにする
    }
    else {                                      //ファイルパスが選択された場合
        String ext = loadPath.substring(loadPath.indexOf('.') + 1);     //ファイルパスのドット以降の文
```

```
字列を取得（拡張子名を取得）
    if (ext.equals("txt")) {                                    //拡張子が「txt」なら
        String lines[] = loadStrings(loadPath);                 //選択ファイルパスのテキス
トを取り込み
        currentPath=loadPath;                                   //現在選択中のファ
イルパスを更新
        println(currentPath);                                   //現在選択中のファイ
ルパスを出力
        F_num = F_num + 1;
        println("num-ok");
        //既存キャラの移動
        for (int i=0; i<F_num-1; i++) {
            Chara_list.get(i).set_up();
            Chara_list.get(i).translate_box();
        }
        println("t-ok");

        //lines に格納された String データをもとに新しいキャラを作成
        Emergent New_Chara = new Emergent(F_num-1, color(round(random(80, 150)),
round(random(80, 150)), round(random(80, 150))), lines[0], lines[1]);
        println("e-ok");
        Chara_list.add(New_Chara);
        println("add-ok");
        P_NUM_R_list.add(0);
        F_clearness_list.add(60);
        mail_frequency_list.add(0);
        F_ELM_list.add(1);
        RM_switch_list.add(0);
        Chara_list.get(F_num-1).Initialize();
        New_switch = 1;
        bg.set_up2();
    }
    //else {                                    //拡張子が「txt」ではないとき
    //println("Not txt file.");                //「txt ファイルではない」と出力
    //}
  }
}
```